VTT Technical Research Centre of Finland

Virtual plants in machine automation research and development

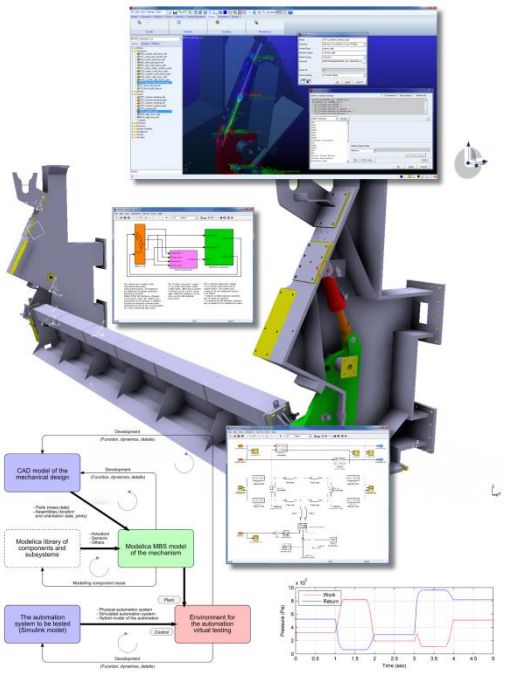
Kortelainen, Juha; Halmeaho, Teemu

Published: 12/02/2014

Document Version
Publisher's final version

RESEARCH REPORT

VTT-R-08126-13



ReUse-R

Virtual plants in machine automation research and development

Authors: Juha Kortelainen and Teemu Halmeaho

Confidentiality: Public

Report's title	
Virtual plants in machine automation research and development	
Customer, contact person, address ReUse-R research project Tekes, Pekka Yrjölä Kyllikinportti 2, P.O. Box 69, FI-00101 Helsinki, Finland	Order reference 40234/11
Project name Prosessinhallinnan suunnitteluratkaisujen uudelleenkäyttö	Project number/Short name 75303/ReUse-R
Author(s) Juha Kortelainen and Teemu Halmeaho	Pages 47/–
Keywords Automation, CAD, control system, hydraulic system, modelling, multibody system, simulation	Report identification code VTT-R-08126-13

Summary

Computational product development has become the mainstream methodology in modern product development. The same trend has been visible also in research, where computational methods have gained popularity beside the traditional approach relying on theory and experimentations. The objective of this project task was to study and demonstrate a realistic approach for an industrial case to reuse existing mechanical design CAD model as the starting point and the template for mechanical system simulation using multibody system simulation, and to use this MBS model as a virtual test plant for automation and control system testing.

In the report, the role of system modelling and simulation in the product process is first discussed and some selected technologies, such as Modelica simulation language and Functional Mock-up Interface specification, are introduced. Then different possible implementations approaches for a test environment of the control and automation system of a multi-technical system are discussed. The latter part of report focuses on describing the selected approach for a demonstration system and its implementation.

The demonstration showed that, at least for the selected case, modelling, simulation and post-processing of a multi-technical simulation system is relatively straightforward and fast with the selected tools. The demonstration gives some understanding of the process for implementing one relatively small multi-technical system but does not give realistic feedback about the challenges in industrial-scale process for virtual prototyping of large and complex systems and related data exchange and data management.

Confidentiality	Public

Espoo 12.2.2014 Written by	Reviewed by	Accepted by
Juha Kortelainen, Principal Scientist	Pekka Rahkola, Senior Scientist	Johannes Hyrynen, Technology Manager

VTT's contact address
VTT Technical Research Centre of Finland, P.O. Box 1000, FI-02044 VTT, Finland

Distribution (customer and VTT)
Pekka Yrjölä, Tekes, 1 copy
VTT, 2 copies

Contents

1 Introduction

Computational product development, including methods such as computer-aided design (CAD), computer-aided manufacturing (CAM), and computer-aided engineering (CAE), has become the mainstream methodology in modern product development. The same trend has been visible also in research, where computational methods have gained popularity beside the traditional approach relying on theory and experimentations [1]. The reason for this is in many cases the cost savings that can be achieved with computational methods. Utilisation of computational methods enables decreasing the use of physical prototypes in the development process. This can reduce costs in two main sectors: first, the direct savings in decreasing the number of built prototypes due to savings in material and work, and second, indirect savings due to shortened design time. Another reason to use computational methods is often forgotten to mention. Modelling and simulation are good means for the designers and experts in the product process to gain understanding about the product under development. The modelling phase helps designers to understand and structure the product and realise the interactions between subsystems and components. Simulating the overall virtual product or its subsystems helps the designers to understand the dynamics and behaviour of the system. All this can be achieved without physical prototypes already in an early phase of the design process. The use of computational methods in product development process helps the designers and experts to design the products according to the technical and project schedule requirements.

This work does not promote any particular software application or an approach to compose a virtual testing system. The selection of the approach as well as the software applications depends on many things, e.g., the requirements for the design system in use and the use of the selected software applications for other purposes. In addition, there can be corporate level policies for using some specific software packages that dictate the selection of tools. Due to this, there is no one optimal solution, but the selection has to be done based on the constraints of the engineering environment.

1.1 Objectives

The objective of this project task was to study and demonstrate a realistic approach for an industrial case to reuse existing mechanical design CAD model as the starting point and the template for mechanical system simulation using multibody system (MBS) simulation, and to use this MBS model as a virtual test plant for automation and control system testing. In the case of the demonstrator, the focus was in modelling, data exchange, and simulation process, and the details and realism of the subsystems, such as the control system, were not emphasised. The emphasis was on modelling and data exchange process so that it would be two-directional when possible. In addition, the openness and standard compliance of the used computational tools and methods were considered as a desirable feature.

The original plan for this project task was to utilise modelling and simulation tools for Modelica modelling language. While executing the research work, this plan had to be updated and a substituting approach had to be selected. This update of the plan explains the structure and contents of this report. In Section 2, the role of modelling and simulation in product process is discussed in general level. This discussion includes also modelling data exchange and data management. In addi-

tion, the Modelica modelling language is introduced and Modelica-related technology, Functional Mock-up Interface (FMI), is discussed. In Section 3, the application of simulation models for machine system control and automation system development and testing is discussed. Some of possible practical modelling and simulation approaches are introduced and their suitability for the case study in this task is discussed in brief. In Section 5, the selected approach and its implementation are described in detail. In Sections 5 and 6, the conclusions and summary of the overall project task are discussed respectively.

2 About modelling and simulation in product process

Complex mechanical products and systems, such as diesel engines and paper machines, involve subsystems from several engineering domains. These subsystems are often designed using domain-specific design methods and established engineering tools. In addition, the design data presentation for different engineering domains differs from each other and it is important for an efficient design process to follow the domain-specific practices also in data presentation. The detailed design of the subsystems is often done by following the given interfaces and requirements, and separate from other engineering domains. On one hand, this approach modularises the design and simplifies the parallel design process of the product. But on the other hand, this approach may lead to ignoring the possible interference effects of the interconnected subsystems. This is especially a risk in systems that are dynamic in nature and which subsystems are strongly coupled. Examples of such systems are e.g. systems that have large accelerations of or large forces acting on their parts, systems in which structural flexibility has remarkable influence on the system behaviour, or systems that have complex control systems that are controlling the dynamics of the system.

System simulation is an efficient mean to master the interaction of subsystems and the overall dynamics and behaviour of the product. The modelling phase of the design process helps the designers and system engineers to structure the product, its subsystems, and components, and to understand the relations between different parts of the system. Simulation of the subsystems and the overall system provides valuable understanding about the interaction of the subsystems and about the overall dynamics of the product. Modelling and simulation also helps the designers and system engineers to communicate with each other and to design the interfaces between the subsystems (Figure 1). All this can be done before any subsystem has been built, either as a prototype or an end product.

While in real products and systems the interfaces of subsystems are fixed and cannot be easily changed, in simulation models this is not the case. In simulation, especially if the modelling and simulation is done using one simulation tool, the subsystem boundaries are often flexible and depends on the preferences of the person who is doing the modelling. From the product design point of view it is important to try to implement the subsystem interfaces according to the real ones, even though it may increase the effort for creating the simulation model. Implementing the same interfaces in the simulation model as in the real system with the same signals and connections not only simplifies the modularisation of the modelling and simulation work but also simulates the interfacing of the real subsystems.

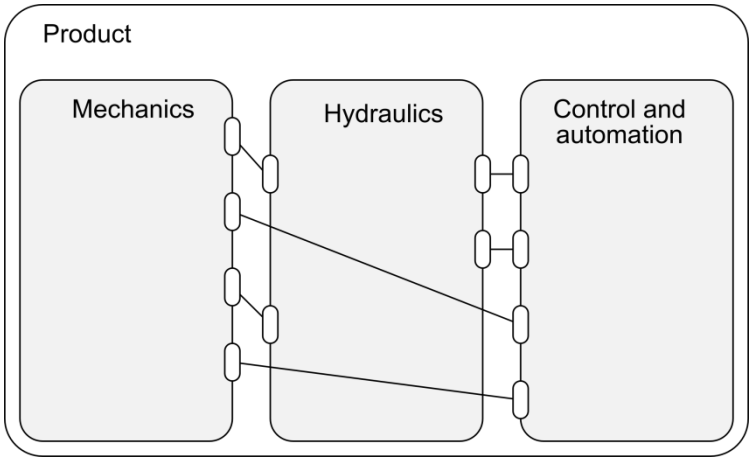


Figure 1: Illustration of the dependencies of the different engineering domain in the design of a product.

2.1 General and application-specific approach

At the same time when the application of simulation in product process is increasing, the concern on data preservability and usability in product life-cycle should be emphasised. More and more product information is stored into different kind of simulation and analysis models, usually in software application specific formats, but very little has been discussed about the usability of this data in later phases of the product life-cycle. The data preservability is briefly discussed in [2], in which a semantic data management approach for one narrow simulation domain is presented. One of the concepts to increase the preservability of the design and simulation data is to separate the valuable data from the tools that are used for e.g. modelling and simulation (Figure 2). Practically this means that the product design data is stored in such a way that the semantics of the data (i.e. the meaning of the data and the used concepts) is known and thus the information the data contains is explicit. This, on the other hand, means that the format for the data is explicitly defined and available. Using standardised data presentation, e.g. applying ISO 10303 (STEP) AP203 and AP214 standards [3, 4] file formats for geometry data, is an example of this. It is always beneficial for data preservability, if the data format is designed and maintained by a party that does not have commercial interests involved.

The present trend of the general data management approach seems to be that the individual software applications for modelling, simulation and post-processing are evolving faster than the development of common data models and standardisation of the data presentation. On the other hand, integration solutions for design and simulation data already exist which fluently integrate design tools (i.e. CAD tools), modelling tools (e.g. FEM pre-processing), simulation tools (numerical solvers), and post-processing tools (data analysis and data visualisation). The common feature of these systems is that they store the data in software system specific format and linking third party software applications with these systems has to be done by following this data format. The data format is specified and maintained by the integration system vendor who has the power to do changes to the format and the data interfaces.

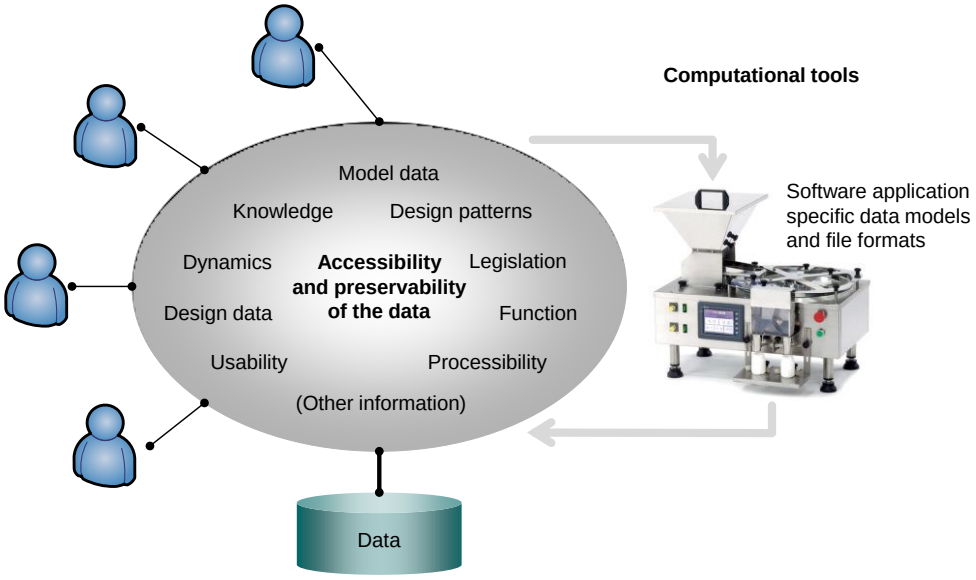


Figure 2: Illustration of the concept of separating valuable product data from the tools to produce or modify it.

The interest in data management and data modelling, and especially increased activity in both research and industrial applications of the Semantic Web technologies, such as Resource Description Framework [5] (RDF) and Web Ontology Language [6] (OWL), have given more focus on the concept of product modelling [7]. The idea of product modelling is to collect all the relevant data into one product model so that the data is linked (from necessary parts) and one piece of information is managed only in one location. The concept of product modelling can be extended to simulation-based product process and simulation data, which eventually can enable the vision of separating the valuable product and design data from the software applications that are used for creating and modifying it. This vision requires still determined research and development work for the concepts and methods, and standardisation for the data models and formats.

2.1.1 Product life-cycle data management

The present trend and fast development of the integration of design tools into large design systems and the retardation of the standardisation of data presentation have already jeopardised the preservability of the data for the whole product life-cycles. It is quite common that the life-cycle of a product in mechanical engineering is 20 years. If the product is in production for five years and the development phase before the production has taken three years, the overall life-cycle of the product is then 28 years. At the same time, the typical life-cycle for a design system in mechanical engineering is about 5 years. Even the computer hardware and computer operating systems have shorter life-cycle than 28 years. The previous commonly used operating system in personal computers, Microsoft Windows XP[1], was released on October 2001 and the expected end of support is on August 2014 [8]. That gives less than 13 years for the life-cycle of this particular operating system. If the product design data is partially stored in a closed, binary format of some simulation tool or a design system, there are no guarantees that the data is usable during the last years of the products life-cycle. On the other hand, if the data was stored in an openly defined format, it is always possible to retrieve the in-

[1] Microsoft XP: http://windows.microsoft.com/en-US/windows/products/windows-xp

formation out of the stored data, even though it may require considerable software implementation effort.

In the area of system simulation, one approach to guarantee the preservability of the information stored in simulation and analysis models is to use an open simulation language to describe the simulation models. There are several simulation languages for system simulation, such as Simscape[2] and Modelica[3]. The Simscape language is designed by Mathworks and is a proprietary product. Modelica is the trademark of the Modelica Association, a non-profit organisation that develops and maintains the specification of the Modelica language. Both of these languages are modern object-oriented languages specifically design for the simulation of complex physical systems. The language specification is available for both of these languages, which means that the information can be retrieved out of the data even though there were no software applications for them available. In the case of Simscape, there are no other software tools at the moment that uses the language. For Modelica, there are several tools available and the use of the language seems to be increasing.

2.2 Modelica simulation language

Modelica is object-oriented language for modelling of physical systems. The language supports all the common object-oriented language features, such as implementation encapsulation, inheritance and subtyping, and is thus well-suited for library development and model data exchange. The language specification is freely available and it is developed and maintained by the Modelica Association [9]. For the use of the Modelica language, a Modelica simulation environment is needed. The environment is used for numerically solving the equations that are defined for the system model in Modelica language. This is an important conceptual feature in Modelica; the language specification is maintained and developed by an organisation that has no direct link to any commercial product that is using the specification. In other words, the Modelica Association is a non-profit organisation and does not have any conflict of interests between the specification and commercial products. It should be noticed that many individual member organisations of the Modelica Association do have direct commercial dependency to the Modelica specification. The commercial independency of the language specification and the tool offering provides better conditions for steady long-term development and maintenance of the language. The investment on the software tools utilising Modelica and the knowhow in the organisation using modelling and simulation e.g. in product development is safe. This is due to open and transparent development of the language and the availability of optional tools for the same simulation language.

The Modelica models are represented in textual, Modelica language form. The models and especially the component connections and dependencies are often visualised as a model graph. The graphical representation of the language is defined in the language specification, which unifies the look and feel of the modelling tools and environments. An example of the graphical representation of a simulation model is shown in Figure 3. The same model in textual form is partially shown in Figure 4.

[2] Matworks Simscape: http://www.mathworks.se/products/simscape/index.html
[3] Modelica Association: https://modelica.org/

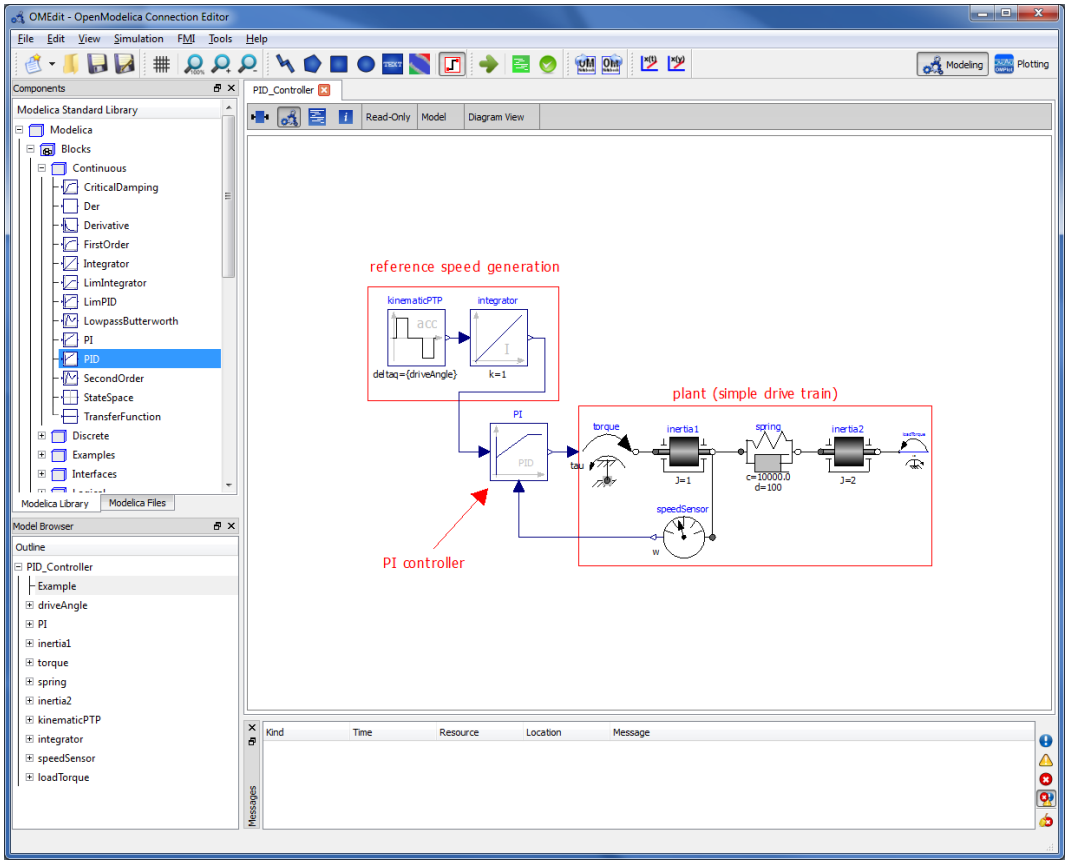


Figure 3: A screenshot of an example model opened (Modelica Standard Library, version) into OMEdit, the graphical modelling editor of the OpenModelica Environment.

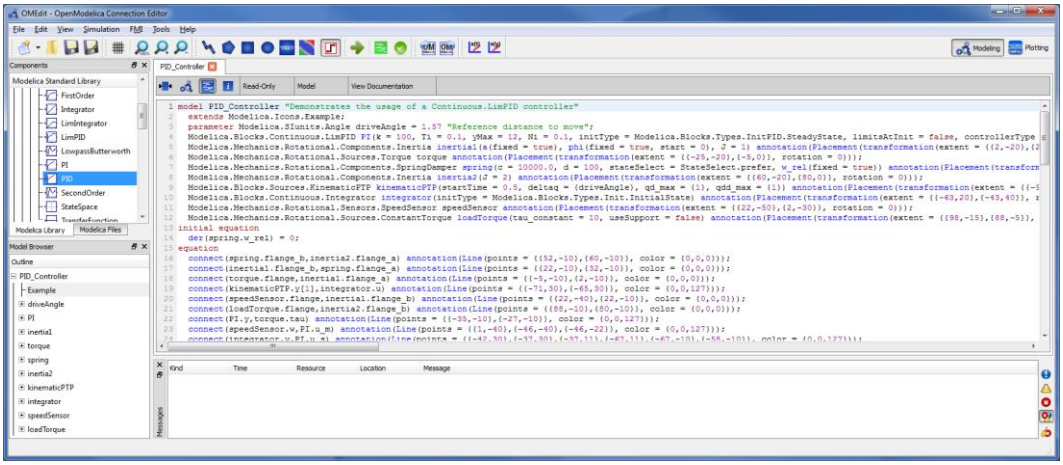


Figure 4: An alternative view in the OMEdit to the same model is in Figure 3.

There are several modelling and simulation tools and environments that utilise the Modelica language, such as [10]:

- Dymola, by Dassault Systèmes
- Vertex, by deltatheta UK Ltd.
- MOSILAB, by Fraunhofer FIRST
- SimulationX, by ITI GmbH
- LMS Imagine.Lab AMESim, by LMS
- MapleSim, by Maplesoft
- MathModelica, by Mathcore

- OPTIMICA Studio for Physical Modeling, by Modelon Ab
- JModelica[4], an open source Modelica environment
- OpenModelica[5], an open source Modelica environment

There are several Modelica environment implementations that work like a compiler for the Modelica language. The Modelica compiler in such an environment gets the model in Modelica language as the input and generates an executable as the output. The executable is a stand-alone software application that includes the description for the specific model together with the numerical solver needed for running the simulation.

The Modelica Association provides a standard library, the Modelica Standard Library (MSL), associated with each specification version. The standard library includes modelling component libraries for many simulation domains, such as mechanical, control, and thermo-fluid systems. The top level sub-libraries of the Modelica Standard Library, version 3.2, are listed in Table 1. For the modelling of especially automation and control systems, the following sub-libraries are available:

- Modelica Standard Library, package *Modelica.Blocks*, including Continuous, Discrete, Logical, and Nonlinear;
- Modelica Standard Library, package *Modelica.StateGraph*;
- There is a new version of the StateGraph library, *StateGraph2*, which is available as a free library for Modelica; and
- *ModelicaDEVS*, a free library for discrete-event modelling using the DEVS formalism.

In addition to the standard library, several free and commercial libraries are available for many areas of system simulation.

Table 1: The top level sub-libraries of the Modelica Standard Library (MSL) version 3.2. [10]

Blocks	Continuous, discrete and logical input/output blocks (**Continuous**, **Discrete**, **Logical**, **Math**, **Nonlinear**, **Routing**, **Sources**, **Tables**)
Constants	Mathematical and physical constants (such as *pi*, *eps*, *h*)
Electrical	Electric and electronic components (**Analog**, **Digital**, **Machines**, **MultiPhase**)
Fluid	Components to model 1-dimensional thermo-fluid flow in networks of vessels, pipes, fluid machines, valves, and fittings.
Icons	Icon definitions
Magnetic.FluxTubes	Components to model magnetic devices based on the magnetic flux tubes concepts.
Math	Mathematical functions for scalars and matrices (such as *sin*, *cos*, *solve*, *eigenValues*, singular values)
Mechanics	Mechanical components (**Rotational**, **Translational**, **MultiBody**)
Media	Media models for liquids and gases (about 1250 media, including high precision water model)
SIunits	SI-unit type definitions (such as *Voltage* and *Torque*)
StateGraph	Hierarchical state machines (similiar power as Statecharts)
Thermal	Thermal components (**FluidHeatFlow**, **HeatTransfer**)
Utilities	Utility functions especially for scripting (**Files**, **Streams**, **Strings**, **System**)

[4] JModelica project: http://www.jmodelica.org/
[5] OpenModelica project: https://www.openmodelica.org/

ModelicaServices	New top level package that shall contain functions and models to be used in the Modelica Standard Library that requires a tool specific implementation.

2.2.1 Requirements-driven development and design, SysML and ModelicaML

The development of the Modelica modelling language and the availability of modelling and simulation tools that are supporting Modelica have provided a fruitful ground for extending the application area of the language. ModelicaML[6] is a UML/SysML extension to combine the benefits of having a graphical system modelling language and simulating the behaviour of the system. The language and the implemented tool support requirements management and simulation-based evaluation of the requirements using Modelica language and tools. Both the ModelicaML language and the related tools are still under research and active development.

2.3 Functional Mock-up Interface version 1.0

Using multiple tools in the modelling and simulation process introduces challenges in reusing the simulation models or model components and co-using numerical solvers, i.e. connecting two or more simulation in runtime. These challenges are common for many simulation domains and have similar features. The Functional Mock-up Interface (FMI)[7] defines a unified, software application independent interface for the exchange of dynamic models and for co-simulation [16]. The FMI specification defines three use concepts:

1) FMI for model exchange [12],
2) FMI for co-simulation [13], and
3) FMI for PLM [14].

In the first concept, FMI is used to exchange model components and/or submodels between software applications, and only one software application is used for running the simulation. In the second concept, FMI is used to define communication between two or more simulation applications (or stand-alone simulation components) and two or more separate solver processes are run in parallel when model components are utilised. In the third concept, mechanisms and interfaces are defined for managing FMI data and related data in product life-cycle management (PLM) systems. The first two of these three concepts are described in more detail in the following sections; the third concept is described only briefly. The development of the FMI concept was started in the European Union funded MODELISAR project that was part of ITEA2 programme[8]. Several research institutes, software providers, and end user companies participated the effort that resulted in defining the FMI specification version 1.0 and providing the necessary supplemental components for the specification. The further development of FMI is organised through Modelica Association Projects (MAP)[9], managed by the Modelica Association. At the time of writing this report, the current stable version of the FMI specification was 1.0.

[6] ModelicaML: https://openmodelica.org/index.php/home/tools/134
[7] Functional Mockup Interface project: https://www.fmi-standard.org/
[8] ITEA2: http://www.itea2.org/
[9] Modelica Association Project: https://www.modelica.org/projects

The FMI concept is based on the interface and behaviour definition between the modelling and simulation software applications and the model components, called Functional Mock-up Units (FMUs). An FMU is a ZIP-compressed file which contains the component and its interface definitions in XML format, necessary functional model data as C source code and/or in binary form as a dynamically loadable library files, and optional auxiliary files for e.g. documentation and providing a component model icon. The internal structure of an FMU ZIP-file is illustrated in Figure 5. The component model data in an FMU is accessed only through C function calls. Because the component model data can be given as a binary form library file, the FMUs can be used for sharing model components without giving the model topology or details in easy-to-read form. This may be the case e.g. when subcontracting is used in product development. The data flow of an FMU is illustrated in Figure 6.

```
// Structure of zip-file of an FMU
modelDescription.xml                // Description of model (required file)
model.png                           // Optional image file of model icon
documentation                       // Optional directory containing the model documentation
      _main.html                    // Entry point of the documentation
      <other documentation files>
sources                             // Optional directory containing all C-sources
      // all needed C-sources and C-header files to compile and link the model
      // with exception of: fmiModelTypes.h and fmiModelFunctions.h
binaries                            // Optional directory containing the binaries
      win32       // Optional binaries for 32-bit Windows
            <modelIdentifier>.dll   // DLL of the model interface implementation

            // Optional object Libraries for a particular compiler
            VisualStudio8           // Binaries for 32-bit Windows generated with
                                    // Microsoft Visual Studio 8 (2005)
                  <modelIdentifier>.lib // Binary libraries
            gcc3.1                  // Binaries for gcc 3.1.
                  ...
      win64       // Optional binaries for 64-bit Windows
      ...
      linux32     // Optional binaries for 32-bit Linux
      ...
      linux64     // Optional binaries for 64-bit Linux
      ...
resources         // Optional resources needed by the model
      < data in model specific files which will be read during initialization >
```

Figure 5: The structure of an FMU ZIP-compressed file. [12]

The FMI concept and specification are software vendor and application independent. This is beneficial for the end users, because it encourages the software vendors to support the specification which increases the number of supporting software applications. A list of software applications that support the FMI specification is kept updated at the FMI website[10]. The current list of FMI capable software applications is given in Appendix A.

2.3.1 FMI for model exchange

The specification for *FMI for model exchange* [12] defines the concrete means to pack a model or a modelling component data into an interchangeable package so that the models and/or model components can be used as model components in other simulation models. The concept of how to use FMI for model exchange is illustrated in Figure 7. FMI enables models, such as control system and controller models, to be exported from one modelling and simulation tool and to be imported

[10] Tools supporting FMI: https://www.fmi-standard.org/tools

into another and used as a submodel component. Any tool that fulfils the FMI specification can be used to produce FMU components or to utilise the FMUs in simulation.

In a FMU component, the model equations are presented either as C source code (which have to be compiled before running the final simulation) or dynamically linkable library component, or a combination of these two. In the target simulation environment, presented in red colour in Figure 7, the submodels are seen as "black box" components and the implementation, structure and hierarchy of the original model are hidden. The FMU component does not include any algorithms needed for solving the component's set of equations, but the numerical solving is done using the target system's numerical solvers. It is possible to generate the C source code for the overall system model containing FMU components, if the C source code is used in the FMU for defining the simulation submodel. Thus, models including FMU components can be used for producing executable code for controllers and embedded systems. Different use scenarios for FMI for model exchange are discussed in more detail in the FMI for model exchange specification document [12].

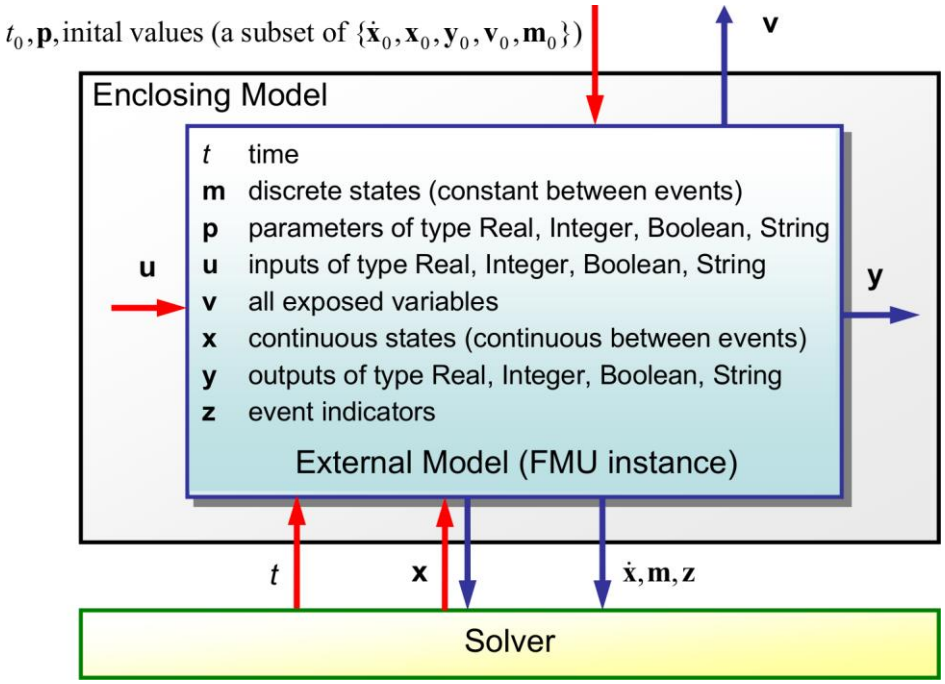


Figure 6: Illustration of FMU data flow. [12]

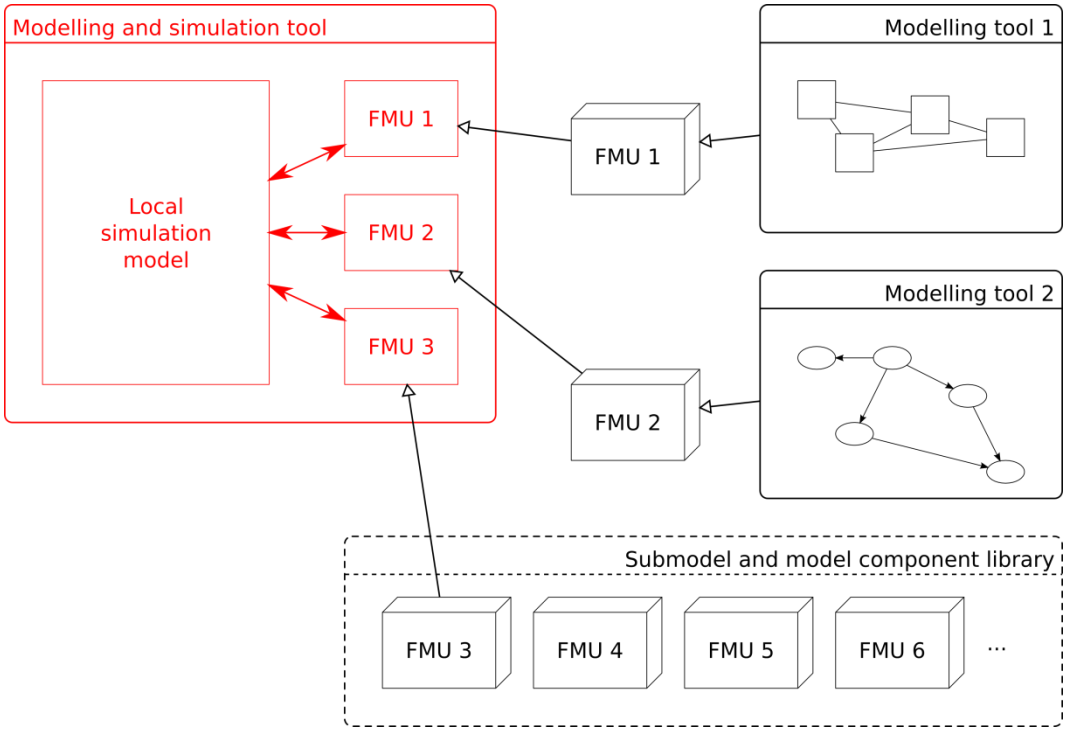


Figure 7: The concept of using FMI components for model or sub-model exchange between different computational software applications.

2.3.2 FMI for co-simulation

The specification for *FMI for co-simulation* [13] defines the means and interfaces for connecting two or more separate simulation tools with their own models to form one simulation. The specification defines two operating modes for FMUs, stand-alone and tool co-simulation:

- FMI for co-simulation stand-alone (this is called *code generation* in [13]); in this mode, the FMI slave dll-component includes a solver for the slave model, thus the FMU is a stand-alone and does not require the original software application to be present for execution (see Figure 8).
- FMI co-simulation tool coupling (this is called *tool coupling* in [13]); in this mode, the slave model is solved with the original software application solver and a specific FMI wrapper is used in between the simulation master and slave processes (see Figure 9).

In the FMI co-simulation, one simulation tool is the master of the simulation and the rest of the simulation system follows it. The specification allows the FMUs to be nested, i.e. a slave of the upper simulation layer can be the master for the lower layer. This concept of using FMI enables connected simulation of different domains in convenient manner, in which e.g. multibody system simulation is connected with hydraulic and control system simulation, and the different engineering domain are modelled separately. This kind of a case is described later in this report in section 4.2 *Complementary approach with MSC Adams and Simulink*, but in that case FMI has not been used for the communication of the simulation tools. This was because the tools that were available and were selected for the demonstration case were not capable for FMI-based simulation.

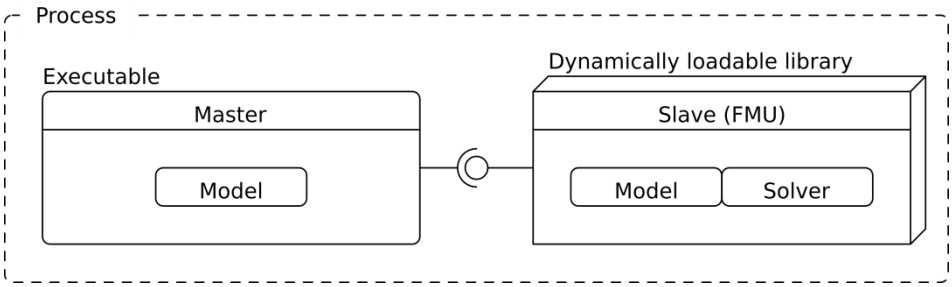


Figure 8: Illustration of the stand-alone mode of FMI co-simulation. In this mode, the FMU contains both the model and the necessary solver routines to simulate the model.

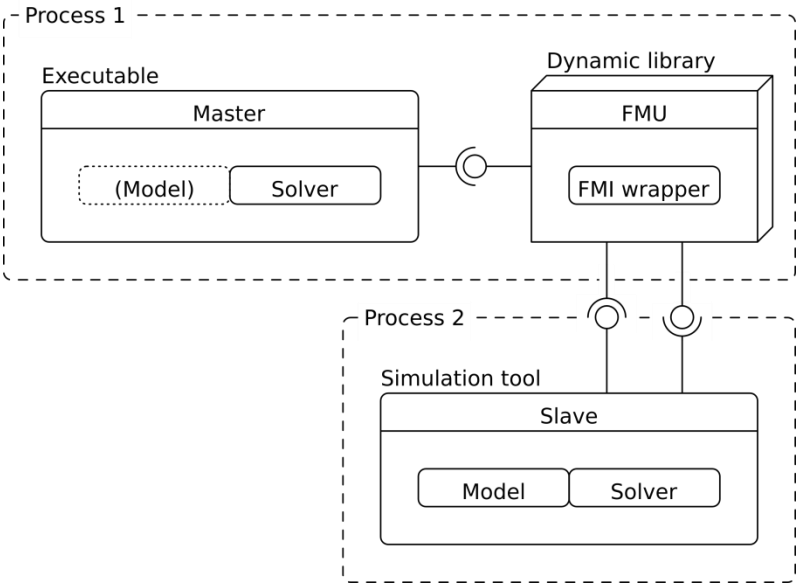


Figure 9: Illustration of the tool mode of FMI co-simulation. In this mode, the FMU contains necessary definitions for the communication between the master process (on the left) and the slave process (on the right). On the slave side, the original simulation tool is needed for running the simulation.

2.3.3 FMI for PLM

The specification for FMI for PLM [14] defines the communication and data exchange practices and details needed for managing FMUs in PLM systems, such as Dassault Systèmes ENOVIA, and to exchange the FMUs between the PLM system and the modelling and simulation applications. The concept is illustrated in Figure 10.

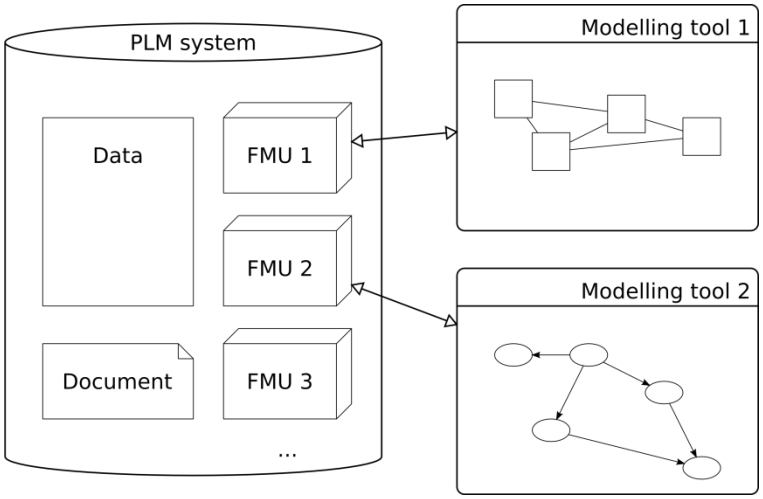


Figure 10: Illustration of the use of FMI together with a PLM system.

2.4 New features in FMI 2.0

The following description of the FMI 2.0 specification is based on the current release of the official FMI documentation [15]. The intention is to point out the most significant changes and improvements to FMI version 1.0 and describe what these changes enable in practice. FMI 1.0 was released 2010, followed by number of beta versions of FMI 2.0. The current version, Release Candidate 1, was published in October 2013 and it will be upgraded into FMI 2.0 after prototype implementations for testing are available. One major conceptual difference in FMI version 2.0 compared to version 1.0 is the merged Model Exchange and Co-Simulation standards. This is also emphasised in the description of FMU by distinguishing Model Exchange and Co-Simulation only by stating the latter to include its own solver in the FMU.

The released specification is mainly intended for software developers using it for implementing FMI functionality for their software. For example communication and function descriptions are given in detail. However, upper level descriptions for end-users can be found in many cases. Performance issues have been considered when designing the new specification, to serve the intention that FMI will also be used in microcontrollers. With large simulation models, performance issues can also arise, although the simulations are run on efficient computers.

Several changes have been made from FMI version 1.0 to version 2.0. Many of them are ticketed as improvements that have been wished by users. In addition, there are some new concepts introduced. The drawback is loss of backwards compatibility to FMI 1.0. Most of the new features are optional, meaning they are not mandatory to be implemented into a tool. This is handled by using capability flags in XML file that tell if the exported FMU is using such an optional feature. The most essential changes are analysed in the following chapters. Compact description of FMI version 2.0 changes is also available as a conference paper from 9th International Modelica Conference held in September 2012 [16].

2.4.1 Merging the documents, schema and header files of Model Exchange and Co-Simulation

In FMI 1.0, there were individual documents for both Model Exchange and Co-Simulation. Now with merged documentation, FMI concept is easier to understand than before. The driving factor for merging the documentation, however, has been combining the overlapping parts of the documents. The working mechanisms in Model Exchange and Co-Simulation have remained untouched in practice, making the descriptions of these two in the previous FMI 1.0 section of this document still valid for FMI version 2.0. In general, the documentation has improved mostly because of better diagrams.

The structure and content of a text file (XML file) that includes all the static information of an FMU is described in the schema files (Figure 11). The XML file contains a model description that consists of different variables and attributes that have a certain value. In the schema files, required data types, default values and value restrictions of these variables are defined. The actual meaning of these variables and attributes are defined in the FMI specification documentation. In FMI version 2.0, there are Common Schema files for Model Exchange and Co-Simulation, but individual fields still exist for both. For example, Co-Simulation has an attribute *canHandleVariableCommunicationStepSize* to describe if the slave is capable of handling variable communication step size.

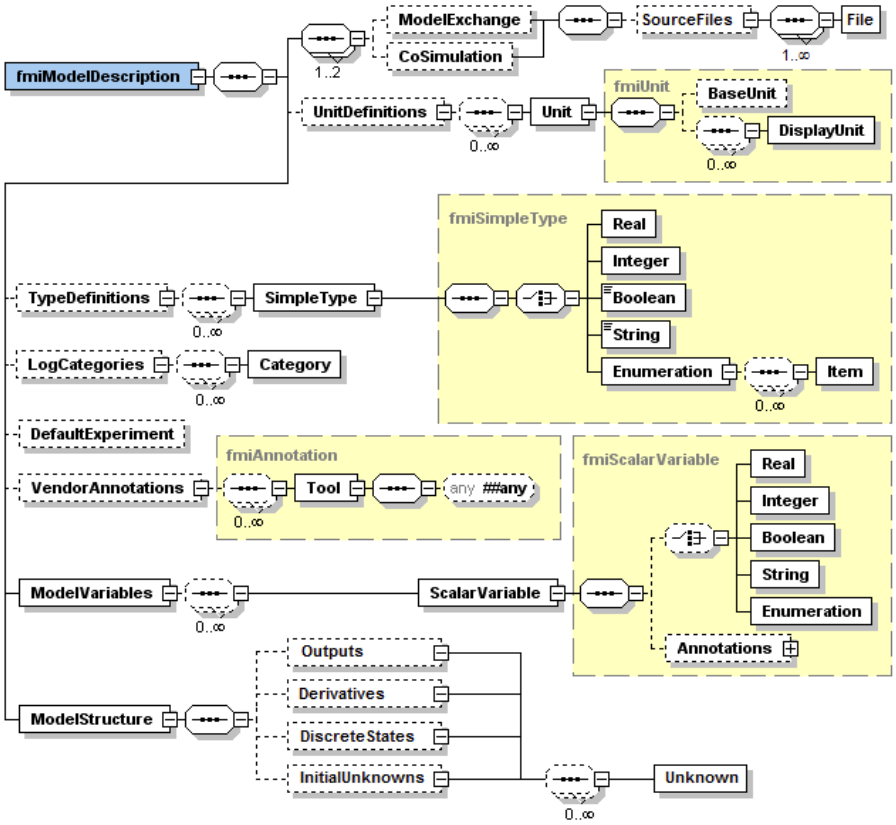


Figure 11: The complete XML schema definition of the FMI version 2.0 [15].

An application programming interface (API) defines the interface to execute functions of an FMU from a program using the FMU. In FMI version 2.0, there is a Common API for Model Exchange and Co-Simulation, but user defined functions

still exist for both. In practice, the API is implemented using C programming language conventions. Common header files include for example all type definitions and common function prototypes but also the specific functions for Model Exchange and Co-Simulation. These header files form the core of the whole FMI concept. A great part of the specification documentation discusses about the content of these files.

2.4.2 New functionalities and flexibility of use

Discrete-time states in the FMU can be optionally defined. This allows for example to linearize discrete-time systems and use the linearized model in linear analysis and synthesis methods. Furthermore, such an FMU may be linearized in every event instant and then the linear model can be used in a model-based controller, or e.g. an extended Kalman filter for nonlinear state estimation. According to Blochwitz et al. [16], the nonlinear model-based control can be implemented by getting the FMU state just before the initialisation and in every sample period, setting new continuous states from an observer, and initialising and getting the FMU state after initialisation. From this state, it is required to perform many simulations that are restarted after the initialisation with new input signals proposed by the optimiser. The nonlinear Kalman filter is achieved by getting the FMU state just before initialisation and in every sample period, setting new continuous states from the Kalman filter algorithm based on measured values, integrating to the next sample instant, and inquiring the predicted continuous states that are used in the Kalman filter algorithm as the basis to set new continuous states [16].

The absolute path to the FMU resource directory is present now also in Model Exchange, in order that the FMU can read all of its resources independently of the "current directory" of the simulation environment where the FMU is imported. For an FMU, it is optional to have any extra resources, but if it does have, they need to be accessible. The resources needed by the FMU, such as maps and tables, are data in the FMU specific files which will be read during initialisation. In addition, more folders can be added under resources (tool or model specific). In order for the FMU to access these resource files, the resource directory must be available in unzipped form and the absolute path to this directory must be reported via argument *fmuResourceLocation* via *fmiInstantiate*. Now that the simulation environment can access the resources outside its "current directory", it can operate as a working hierarchy master which picks up the FMUs generated by the working hierarchy slave which may not have access to master's "current directory". For example in scenario where simulation environment is a tester that tests FMUs produced by the slave, it is possible for the slave to generate test items and prepare them for testing independently of the master.

In FMI version 2.0 compared to version 1.0, variables exposed by the FMU are categorised in different way by attributes "causality" and "variability". Attribute "causality" is an enumeration that defines the causality of the variable. Allowed values are parameter, input, output, and local. For value "parameter" it applies: an independent variable that must be constant during simulation. When "input", the variable value can be provided from another model. When "output", the variable value can be used by another model. For value "local", it applies that the variable is calculated from other variables and it is not allowed to use the variable value in another model. Attribute "variability" is an enumeration that defines the time dependency of the variable, in other words it defines the time instants when a varia-

ble can change its value. Allowed values are constant, fixed, tunable, discrete and continuous. When “constant”, the value of the variable does not change. When “fixed”, the value of the variable is fixed after initialization. For value “tunable” it applies that the value of the variable is constant between externally triggered events due to changing variables. In this case, the attribute “causality” needs to be “parameter” or “input”. For value “discrete” it applies that the value of the variable is constant between internal time-, state-, and step-events defined implicitly in the FMU. For value “continuous”, no restrictions exist on value changes.

The new value, “tunable”, for variability introduced in FMI version 2.0 allows a modelling environment to expose independent parameters that can be manually “tuned” during simulation. “Tuning a parameter” during a simulation does not mean to “change the parameter online” during simulation. Instead, this means: 1) Stop the simulation at an event instant; 2) Change the values of the tunable parameters; 3) Compute all parameters that depend on the tunable parameters; and 4) Resume the simulation using as initial values the current values of all variables and the new values of the parameters. Changing the parameter values of an FMU during simulation is possible using specific Set-functions. These parameters should be defined beforehand as “tunable” variables. Therefore, the software doing the exporting of an FMU should have an option for the user to enable tunability for desired parameters. For inputs, changing parameter values is possible without tunability attribute. Using the tunability, parameter values of an FMU can be changed outside the FMU. For example in the Model Exchange mode, the simulation environment can change the values of an imported FMU model obviating the change of these parameters (in the software application that was used for creating the model) and exporting and importing it again. With hierarchical FMUs, all variables in an external FMU that shall be visible and/or accessible from the environment need to be “exposed”, in other words in the root-level FMU a corresponding variable needs to be defined and in the generated code this variable must be assigned to the corresponding variable of the external FMU. As a result, only variables from the top most FMU are visible or accessible from the environment where the FMU is called.

2.4.3 Performance and reliability upgrades

Connected signals can be checked to have match in units. In addition, the variable values can be changed to match by performing unit conversion for same physical quantity. This is enabled with the improved unit definitions. In FMI version 2.0, the unit names are expressed by using the seven SI base units together with SI derived unit “rad”, instead of using standardized unit names which has been problematic in FMI version 1.0.

An FMU has an internal state consisting of all values that are needed to continue a simulation. This internal state consists of the values of the continuous states, discrete states, iteration variables, parameter values, input values, file identifiers, and FMU internal status information. The complete FMU state can be saved, restored, and serialised to a byte vector that can also be stored into a file. As a result, a simulation can be restarted from a saved FMU state. This applies for both Model Exchange and for Co-Simulation. Rejecting steps in variable step-size Co-Simulation master algorithms, is now performed by saving and restoring the state instead of the less powerful method of the FMI version 1.0.

Support for more sophisticated Co-Simulation master algorithms (e.g. variable step sizes, higher order signal extrapolation etc.) that control the data exchange between the subsystems and the synchronisation of the simulation solvers of the slaves is added. The master algorithm itself is not part of the FMI standard.

In FMI version 2.0, the dependency information of the outputs is stored in the XML description. This can be used for the detection of algebraic loops when FMUs are connected with other parts of the model. Artificial or "real" algebraic loops over connected FMUs can be handled in an efficient way also in Initialization and Event Mode (discrete time). In FMI version 1.0, algebraic loops in Initialization and Event Mode could not be handled.

Directional derivatives can be computed for derivatives of continuous-time states, for discrete-time states, and for outputs. This is useful when connecting FMUs and the partial derivatives of the connected FMU shall be computed. If the exported FMU performs this computation analytically, then all numerical algorithms based on these partial derivatives (for example the numerical integration method or nonlinear algebraic solvers) are more efficient and more reliable.

2.4.4 Future of FMI usage and progress of the simulation tools

In addition to the mentioned upgrades, a number of minor improvements have been done to increase the efficiency of the FMI standard. In spite of being relatively young, the FMI standard has enjoined attention of many simulation software designers and users. This has helped to gather extensive amount of user experience for the basis of the development.

FMI is used in many companies with good results, which drives the simulation software vendors to include FMI in their products. If they do not, their software will be lagging behind from the general progress of simulation tools. The support for FMI version 1.0 is already available for many simulation software applications although the standard has been out from year 2010. This is the result of the demand from the big companies to exploit FMI in their product design and production.

3 Virtual automation test environment

An ideal work process and data flow for the mechanical system simulation model generation for automation testing is illustrated in Figure 12. In this description, a Modelica tool (such as Dymola) is assumed for modelling and simulation for the mechanical system and Simulink for the automation and control system. The environment for the automation virtual testing is composed of two main modules: the mechanical system (plant) and the automation system (control). The process for creating the mechanical system simulation model begins from the design model of the mechanical system (CAD model, the upper left corner rectangle in the figure). This model contains the necessary information about the individual parts of the system (geometry, mass, centre of mass and mass inertia) and the assembly information (location and orientation of the parts, and location, orientation, and the type of the joints connecting the parts). The design model of the mechanical system is obtained from the design system, such as CATIA. In the next phase, the assembly model is complemented with the actuators, sensors, and elastic compo-

nents (springs and dampers) to form the multibody system model of the mechanical system (the green rectangle in the centre of the figure). In addition, the interface for the automation system interaction is designed and implemented. The interface contains the plant outputs for sensor signals (measurement signals of the mechanical system) and the plant inputs for controlling the actuators. Existing library components are used in this phase, when possible (the dash-lined rectangle on the left side of the figure). Newly defined components (e.g. new actuators) are added to the component library for later use. In the final phase, the mechanical system simulation model is connected with the automation system simulation model (on the lower left corner of the figure), and the overall system is ready to be used for automation system testing (on the lower right corner of the figure).

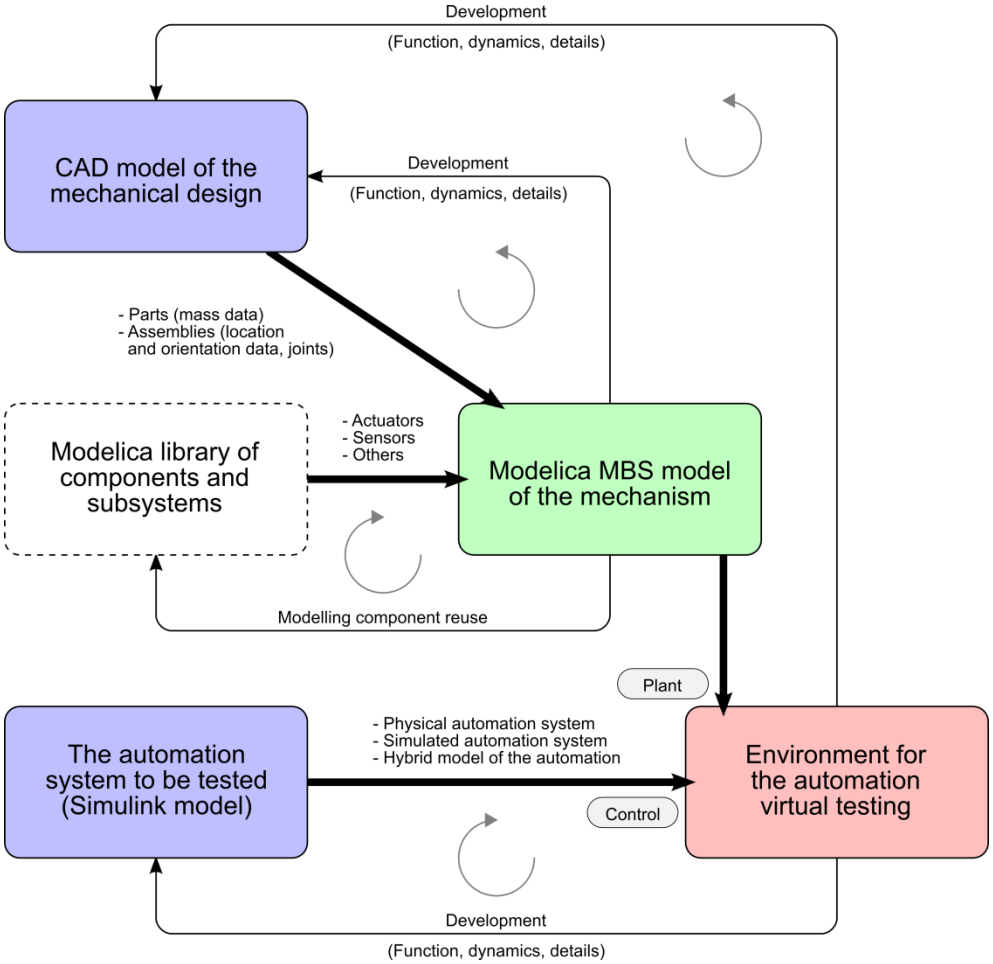


Figure 12: An ideal work process and data flow for the demonstration.

The creation of the environment for the automation system virtual testing is straight forward, when the overall design of the system is followed and the simulation models for both the mechanical system and the automation system are created following good system modelling principles. This means that the physics and logics of the systems are modelled correctly and the implementation of components, such as measurements in the mechanical system, are realistic. The difficulty of managing the quality of a simulation model is typically strongly related to the complexity of the model. Finding modelling and logical errors from a large and complex model is demanding due to the large number of model components, but also because the symptoms of a faulty model may not be obvious. It is always a good practice to test each modelled subsystem separately to minimise the risk of

modelling or logical errors, before combining the overall simulation model. The importance of good documentation cannot be over emphasised. Following strictly the system design in the model implementation minimises the need for parallel model documentation.

Application of the virtual environment for the automation system testing in real industrial product process is challenging. In this work, the actual application of the virtual testing environment is not demonstrated. The selected scenario for the use of the environment is for testing the designed automation system. In principle, using simulation in product development process for testing and validating the design is not the optimal approach. In the design process, the investment on the virtual prototype for validating the design does not feed any added information back to the design process during the early design phase but after most of the design work has already been done. This means, the valuable information about the plant and the automation system interaction cannot be exploited in the design process. If a design flaw is discovered in virtual testing, the design work has to take steps backwards to correct the issue. If the simulation was used already during the design process, this step could have probably been omitted. The information feedback in different design loops, either in form of useable knowledge or more concrete in form of reusable design or modelling components, is illustrated with grey arrows in Figure 12.

3.1 Target system and its subsystems

A relatively simple mechanism was selected for the target system of the automation and control system virtual test environment. The target system is a partial pick-up mechanism in the Metso Paper OptiPress system (Figure 13). The modelled mechanism does not follow exactly the real system but some details of the mechanism have been modified and some parts have been left out. The objective of this study was to demonstrate the process of using an existing CAD model as the starting point for building a virtual test environment for system automation and control. A CAD model image of the modelled and simulated target system is presented in Figure 14.

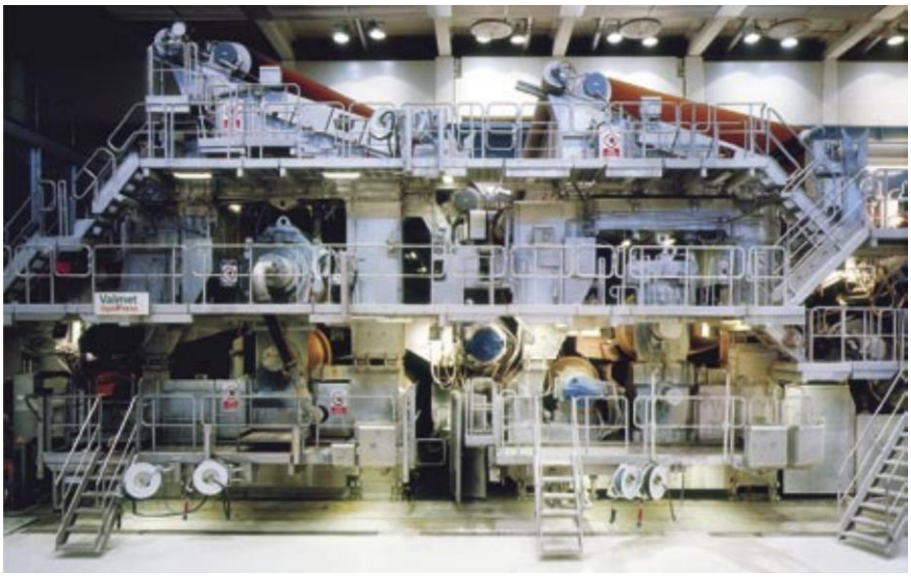


Figure 13: The Metso OptiPress system [17].

In the target system, the pick-up function gets the paper web from the forming section of the paper machine and feeds it through the press section and later through the drying section. In a real system, the control and automation of the mechanism has many functions and critical timings, but they are not included into this demonstration. In the demonstration case, only a simple lowering and lifting operation of the pick-up blade was modelled and simulated.

The modelled and simulated mechanism consists of frame structures (the grey parts in both ends of the system in Figure 14), levers (the green parts in Figure 14), hydraulic cylinders (the red and orange parts in Figure 14), and a pick-up blade (the grey part in the middle of the system). The hydraulic cylinders operate the levers that turn around their pivot joint (in the right lower corner of the lever in Figure 14) and make the pick-up blade to move down and up. The hydraulic cylinders are driven by the hydraulic subsystem, which is controlled by the control subsystem. The dependencies of the target system's subsystems are illustrated in Figure 15.

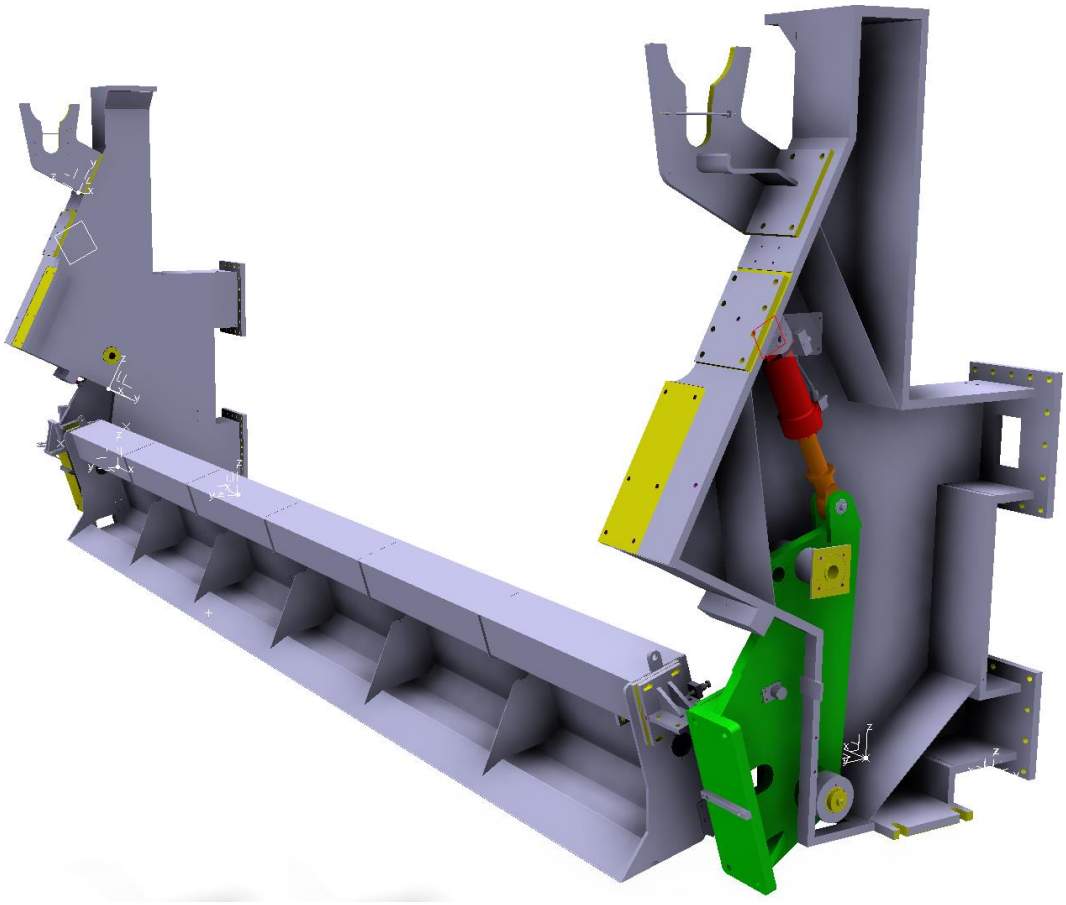


Figure 14: A CAD image of the target system, a pick-up mechanism of the Metso OptiPress system. In the picture, the side plate of the frame structure has been removed.

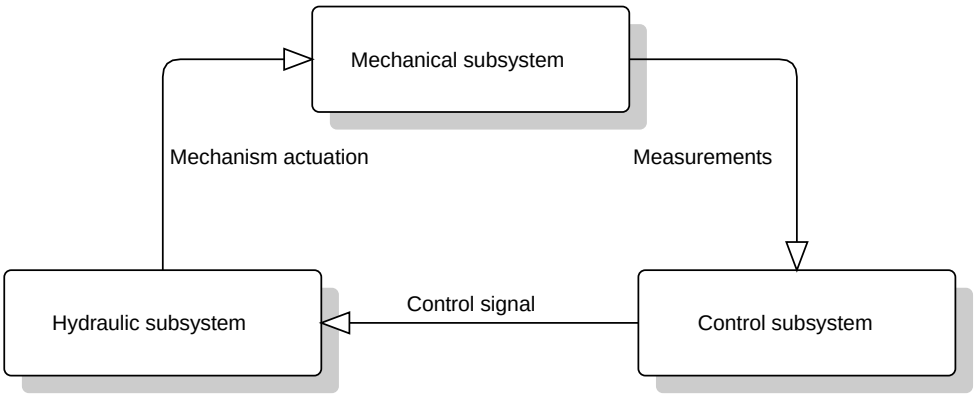


Figure 15: Target system's subsystems and their dependencies.

3.2 Approaches for the implementation

The necessary tool chain and the process described in a principle level in the previous section can be implemented with several different kinds of strategies and software applications. A fully integrated design and simulation environment can be applied, in which the data flow between different modelling and simulation tools is often seamless. The drawback in this approach can be the availability of the integration of some special software application components and the number of options for each modelling and simulation purpose. Another strategy is to compose the overall system using separate tools for each phase in the process. This approach offers the flexibility of selecting the most appropriate tools for each phase, but often requires additional work for connecting the tools in the process to enable seamless data flow between different software applications. The following optional solutions for the implementation were considered:

- CATIA V6 environment with mechanical design and the systems engineering modules; this approach provides in principle good software component integration and data flow in the modelling and simulation process, but the development of the required software features are still in progress at the time of writing this report,
- CATIA V5 together with LMS Virtual.Lab Motion software package; this approach is based on the previous version of the CATIA software application and a third-party simulation tool, LMS Virtual.Lab Motion, which is implemented on top of CATIA V5 software platform,
- A general CAD software application with a general MBS software application and a general hydraulic and control system simulation software application; this approach provides flexibility in individual domain software application selection, but may introduce challenges in data exchange,
- A CAD software application with Mathworks Simscape/SimMechanics, Simscape/SimHydraulics and Simulink; this approach limits the number of different software application into two and would provide good software integration between different simulation domains.

These options are discussed in more detail below.

3.2.1 CATIA V6 environment

There are numerous commercial tools that can be used for creating a virtual automation testing environment. The Dassault Systèmes V6 (version 6) architecture, 3DEXPERIENCE, consists of the CATIA V6 design environment, ENOVIA data

management system, DELMIA manufacturing modelling and simulation environment, SIMULIA detailed physics simulation environment, and a set of other software components. ENOVIA provides the PLM functionality for the architecture and can be seen as the backbone of the architecture together with the V6 platform. The SIMULIA environment is built around the Abaqus finite element method software package. Abaqus can be used for structural analysis, computational fluid dynamics, and other multi-physics applications.

CATIA[11] V6 design environment offers a solution for fully integrated data flow and unified user interface for all the modelling and simulation modules. The environment has tools for e.g. part design (design of mechanical components), assembly management, and system simulation. The system simulation module, based on the Dymola[12] system modelling and simulation package, has an interface to some external simulation tools, such as Matlab/Simulink[13], and it can be used together with hardware-in-the-loop tools, such as xPC Target[14] and dSPACE[15].

International Organization for Standardization standard ISO/IEC 15288 [22] defines concepts and general processes for systems engineering (SE) and system life-cycle process. A more verbose and detailed description of the SE process is given by International Council on Systems Engineering (INCOSE) in its Systems Engineering Handbook [23]. The CATIA V6 design environment follows the concept of the SE process. The platform has tools for definition and linking of requirements (requirement engineering), design and definition of system functional architecture, design and definition of system logical architecture, and design of the physical properties of the product (CAD).

From the above described options, the CATIA V6 environment was originally selected for this demonstration. This was due to the long-term design objectives of the CATIA V6 environment, which are planned to support simulation-based product development and also the simulation-based product life-cycle process. At the time of implementing the demonstration system, the available version of the CATIA V6 software was 2011x.

3.2.2 CATIA V5 and LMS Virtual.Lab Motion

The previous version of the Dassault Systèmes design environment, CATIA V5 (version 5), offers another approach for an integrated design and simulation environment. A third party simulation software package, LMS Virtual.Lab[16], is built on the CATIA V5 software platform. The Virtual.Lab modelling applications utilise the part and assembly modelling capabilities of the CATIA environment, and thus fully integrate with the CATIA V5 environment and its data flow. LMS Virtual.Lab Motion, the module for simulating the dynamics of mechanical systems, can be used for co-simulation with external simulation packages, such as LMS Imagine.Lab AMESim[17] and Matlab/Simulink. LMS Virtual.Lab has software interfaces to some other CAD packages, such as Pro/Engineer, CATIA V4, and Au-

[11] Dassault Systèmes, CATIA: http://www.3ds.com/products/catia
[12] Dassault Systèmes, Dymola: http://www.3ds.com/products/catia/portfolio/dymola
[13] Matworks, Simulink: http://www.mathworks.se/products/simulink/index.html
[14] Matworks, xPC Target: http://www.mathworks.se/products/xpctarget/index.html
[15] dSPACE software: http://www.dspaceinc.com/en/inc/home/products/systems.cfm
[16] LMS Virtual.Lab software: http://www.lmsintl.com/simulation/virtuallab
[17] LMS Imagine.Lab software: http://www.lmsintl.com/imagine-amesim-1-d-multi-domain-system-simulation

todesk Inventor, and CAD models in formats, such as STEP, IGES, and ParaSolid, can be read into the system.

The simulation environment approach of LMS Virtual.Lab is tightly integrated with the CATIA V5 platform. The geometry modelling features rely on CATIA 3D modelling features and the system has excellent and seamless data flow from CATIA V5 CAD modules. On the other hand, the openness of the solution, ability to connect third party and in-house software applications to the process, and extensibility of the modelling and simulation capabilities are an open question.

3.2.3 General CAD and general MBS

Many multibody system simulation software packages can read CAD models in either standard formats, such as STEP (ISO-10303, AP203 and AP214) and IGES, or in some proprietary geometry formats, such as ParaSolid and ACIS. This approach enables flexible selection of the software applications in the process. The challenge in this approach is the implementation of the geometry import of a CAD model into the MBS software application. If the MBS software application does not support full solid geometry import, but converts the geometry into a faceted surface representation, the geometrical features, such as centre points of spheres and centre lines of cylinders, are not available for the modelling in the MBS software application. This may become a problem, if the mechanism is complex. In addition, if the solid geometry import is not successful due to inaccuracies in the geometry surface representation, the imported part does not form a solid (i.e. the volume defined by the surface facets is not closed) and the mass properties for the part cannot be defined based on the geometry and given density.

There are available several commercial, general-purpose multibody system simulation packages. Often mentioned software applications are:

- LMS Virtual.Lab Motion,
- MSC Adams,
- Recurdyn, and
- Simpack.

The approach described above was selected for the implementation of the demonstrator.

3.2.4 A CAD software and MatWorks Simscape/SimMechanics

MathWorks SimMechanics is a modelling library in the Simscape modelling environment that is designed for three-dimensional mechanical system simulation. The Simscape[18] language itself is similar to Modelica simulation language. The language is based on the MATLAB language and extends the Simulink environment with modelling libraries especially for physical systems. The Simscape language is designed and maintained by MathWorks Inc. The Simscape basic library contains components for one-dimensional translational and rotational mechanics, electrical systems, hydraulic components and systems, and thermal systems. There are extended libraries for simulation of

- multibody system (SimMechanics)
- drivelines (SimDrivelines)

[18] MathWorks Simscape: http://www.mathworks.se/products/simscape/

- electronic and electromechanical systems (SimElectronics)
- hydraulic systems (SimHydraulics)
- electrical power systems (SimPowerSystems)

The Simscape models and components can be mixed with components and models in Simulink and MATLAB.

The most relevant package of Simscape for this work is the SimMechanics that provides functionality for multibody system simulation. There is an additional product, SimMechanics Link, that enables data exchange between Pro/Engineer, SolidWorks, and Autodesk Invertor CAD systems and Simscape.

4 Implementation

4.1 Original approach with CATIA V6 and Modelica

The CATIA V6 platform is a large and complex software package and e.g. the installation, including the software documentation, on the Windows 7 64-bit platform requires about 4.7 GB of disk space. In addition, the tested version of the software was relatively early in the latest CATIA V6 series and many features of the system were clearly still under development and some important features were missing. Also, the stability and performance of the system needed some improvement. Due to all these, learning to use the system in intended way was a big challenge and it is obvious that some of the negative user experiences are because of the familiarising process was still in progress.

In the tested version of CATIA V6, the integration of the Modelica-based system simulation module, *Dynamic Behavior Modeling* (DBM, Figure 16), was still partial. This system module was one of the most important ones for this project task, because it was meant to be used for modelling both the dynamics of the mechanical system and the connected control system. One of the most limiting features was that the model data that existed in the CAD modelling modules of the CATIA system, *Mechanical Design* (MDE, Figure 17), was not available for system modelling. In the case of mechanical system modelling, this was especially problematic, because for MBS simulation the fundamental information for the system is the mass, centre of mass location and mass inertia tensor for the system parts (i.e. mechanical system bodies). In most of the commercial MBS software applications this information is provided automatically based on the geometry and density information of the parts. Another feature for which the system geometry is used is to define the location and orientations of system parts, joints and forces. In the CATIA V6 DBM module, all these definitions had to be done manually copying the information from the MDE module.

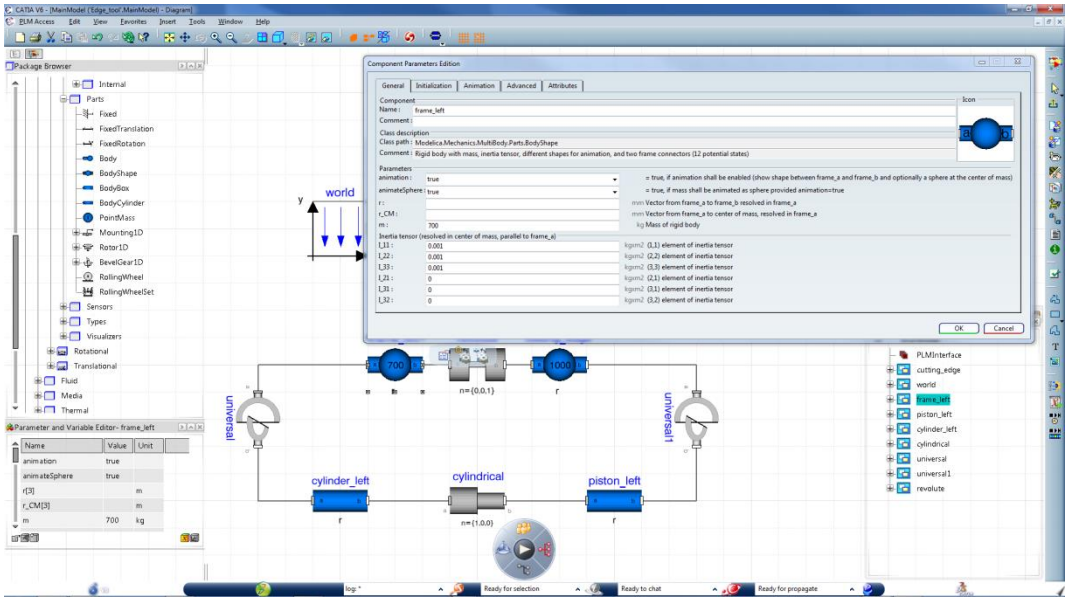


Figure 16: An example of an attempt to create the MBS model of the cutter system in the CATIA V6 Dynamic Behavior Modeling module.

As the MBS modelling is highly related to the geometry of the mechanical system, 3D modelling of the mechanism is the convenient approach. 2D graph-based modelling view gives explicit and clear information about the topology of the system and the connectivity of its components, but does not give any feedback about the location and orientation of the components. 2D and 3D views complete each other and increase productivity in modelling and simulation process, but typically most of the modelling work is done in 3D. In the CATIA V6 DBM module, the modelling of a 3D mechanism is done using 2D graph-based user interface.

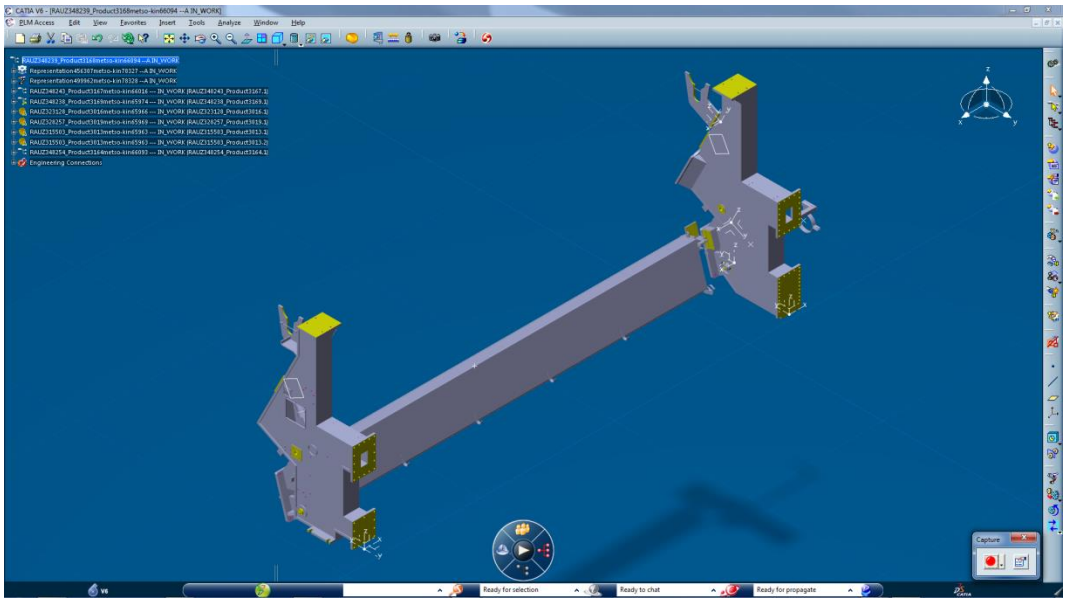


Figure 17: The CAD model of the cutter system in the CATIA V6 Mechanical Design module.

4.2 Complementary approach with MSC Adams and Simulink

Due to limited resources in this project for the modelling and simulation, the virtual test environment of the mechanical system for control and automation testing

was decided to be implemented using the combination of MSC Adams and Matlab/Simulink software applications. The main reason for this selection of the software was that the author had previous experience on these software applications and their application on similar simulation tasks. It is important to notice that the implemented modelling and simulation process does not demonstrate the bidirectional data flow and iterative nature of the process as illustrated in Figure 12. The MSC Adams software, used for the mechanical system simulation, can utilise the 3D solid geometry that was available from the CAD system, but there is no direct link from the MBS simulation back to the CAD design system. This means, if there is need for modifications in the mechanical system design, the model has to be updated manually and the change requests from the mechanical system simulation have to be exchanged by other means.

4.2.1 Mechanical system of the test case

The work for implementing the virtual test environment started with writing out the existing CAD model from the CATIA V6 environment in STEP format (ISO 10303-21, AP203). The STEP model was then converted into commercial Parasolid format (format version 19.0). The MSC Adams View pre-processing software application uses natively Parasolid as its geometry format. With this format, it was possible to import the whole system assembly at once and the parts of the assembly retain their original mass properties. The model did not save its assembly hierarchy in the conversion process, but the whole assembly was flattened when imported into MSC Adams View pre-processor. The imported CAD model assembly in MSC Adams View processor is shown in Figure 18 with partial model part list visible on the left side and mass properties for one part shown in the Information window.

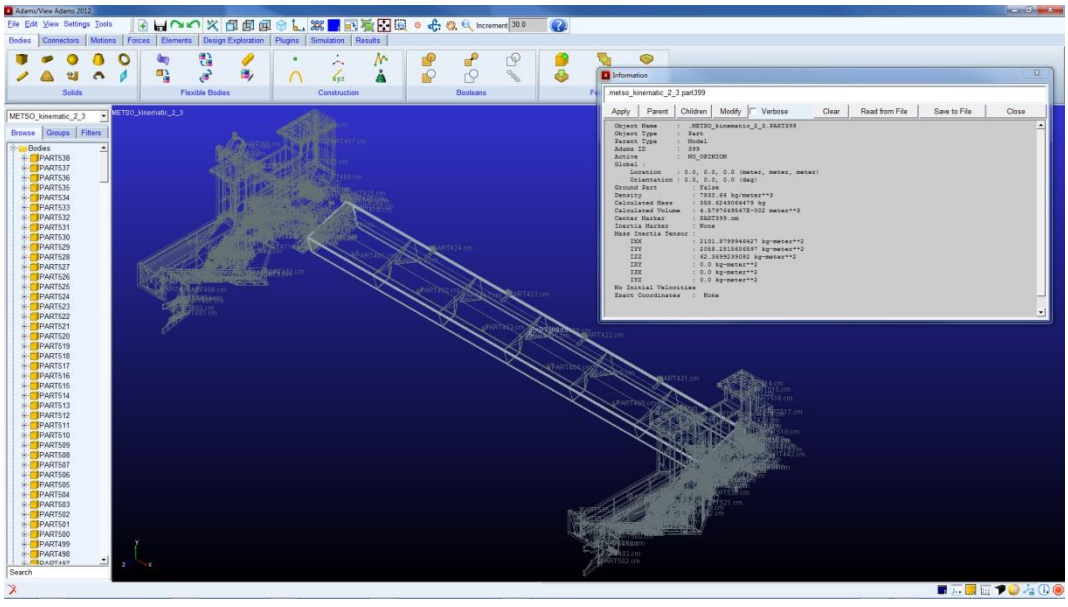


Figure 18: The assembly of the test case imported into MSC Adams using Parasolid format; the mass properties of one part in the assembly are shown.

The next phase in the modelling process was to rebuild the assembly and join individual parts to form rigid MBS bodies suitable for the simulation of the mechanism (Figure 19). The motivation for doing this is computational. In the multibody system formulation used in the MSC Adams software application, each multibody system free part (body) introduces 18 equations into the set of equations to be

solved. On the other hand, each multibody system part may contain several geometries (e.g. assembly components) that are treated as one rigid body in the simulation. In the selected modelling approach for the demonstration case there are 11 individual rigid moving parts in the system and the system ground part.

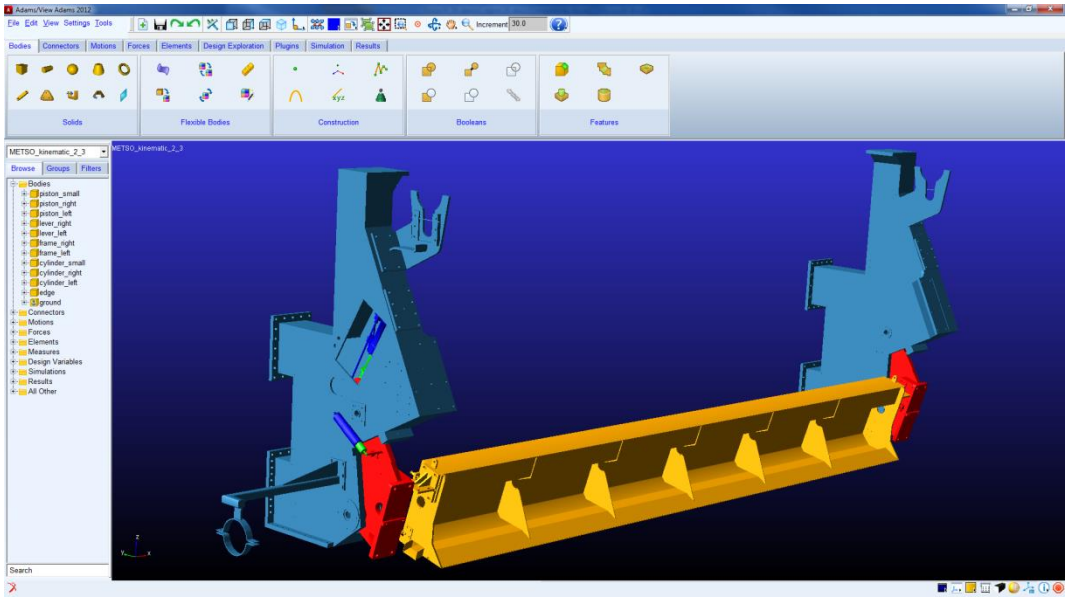


Figure 19: The parts of the CAD model have been joined to form MBS model bodies (the opened list on the left side of the screen view).

After the suitable bodies for the MBS model were formed, the model was ready for defining the joints, actuators (forces representing hydraulic cylinders and dampers), and other modelling components. Due to the use of Parasolid solid geometry, the locating of the model components was straightforward. The locations and orientations of the model components could be defined using geometry features, such as centre points and corners. In Figure 20 is shown the location of the joint for the piston end of one hydraulic cylinder (highlighted both in the geometry view and in the component browse list on the left). In this case, a hooked joint (cardan joint) was used to prevent the cylinder piston to unnecessarily rotate around its own axis.

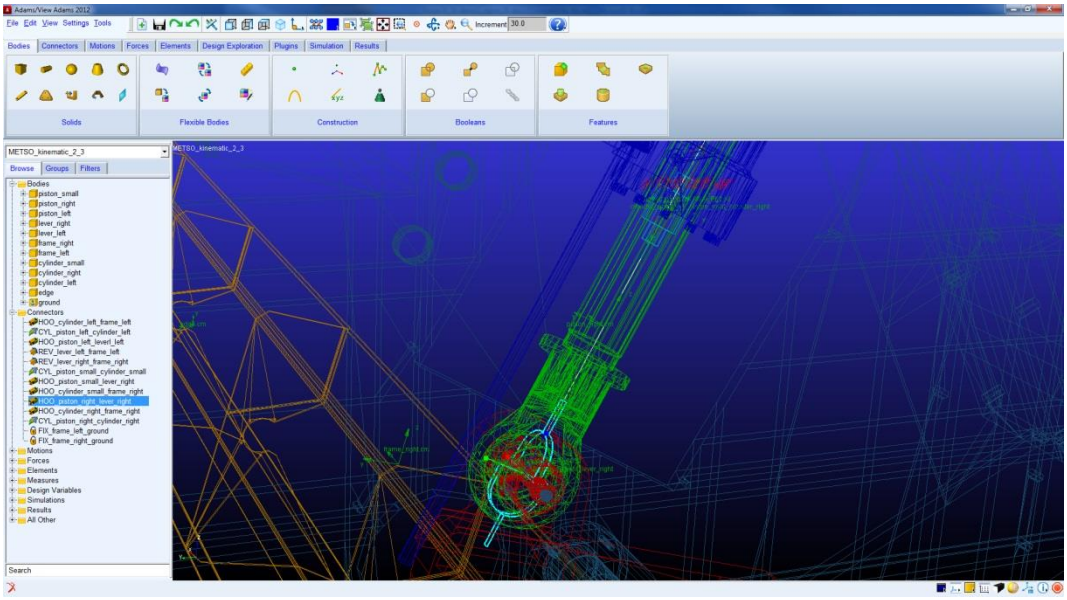


Figure 20: Location and orientation of the piston end of one of the hydraulic cylinders in the MBS model.

The force components, representing e.g. the hydraulic actuators and cylinder end contact forces, were defined with a similar manner as the joints (Figure 21 and Figure 22). The use of geometric features, such as centre points, simplified the modelling process and made it fast.

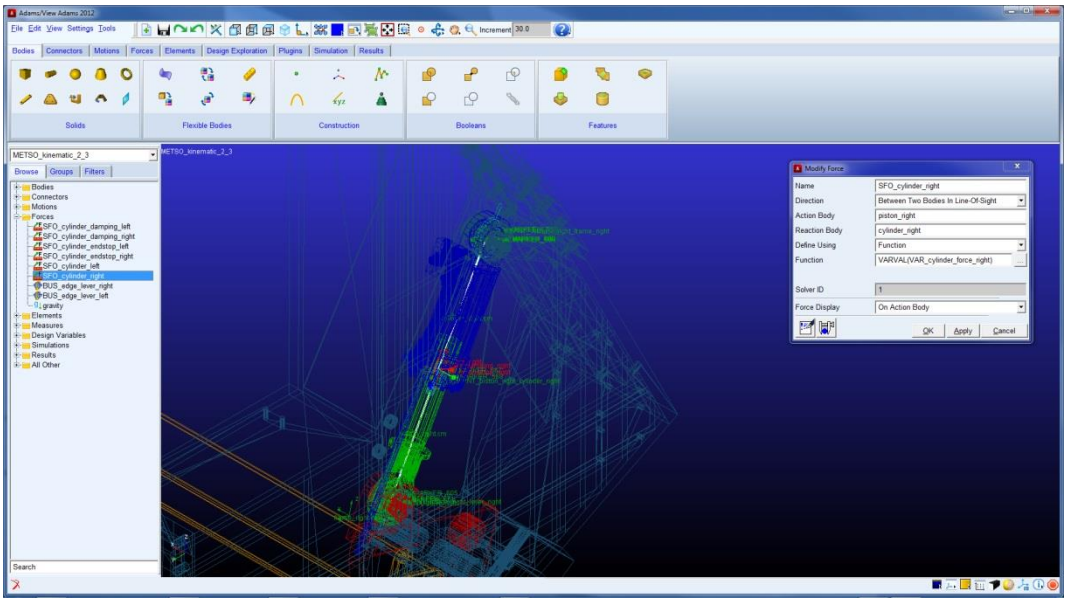


Figure 21: Modelling of a hydraulic cylinder actuator force component. A single component force (force acting between two points in space) was used for the hydraulic cylinder force.

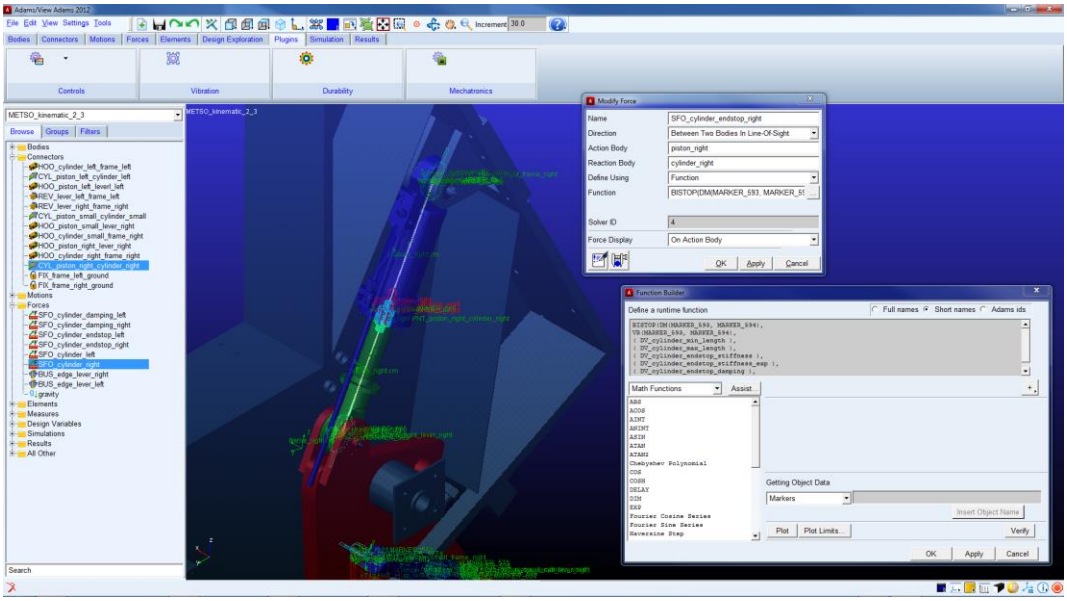


Figure 22: Modelling of a hydraulic cylinder end stopper contact force. Standard force functions were used for modelling the force components in the case.

The mechanical model of the test system can be considered to be a relatively simple multibody system model and it contained the following 11 bodies:

- Hydraulic cylinders (`cylinder_left` and `cylinder_right`)
- Damping cylinder (`cylinder_small`)
- Cutting edge (`edge`)
- Cutter frames (`frame_left` and `frame_right`)
- Cutter levers (`lever_left` and `lever_right`)
- Hydraulic cylinder pistons (`piston_left` and `piston_right`)
- Damping cylinder piston (`piston_small`)

In total, the model included the following 13 joints:

- Cylindrical joints between hydraulic cylinder and piston (`CYL_piston_left_cylinder_left`, `CYL_piston_right_cylinder_right`, and `CYL_piston_small_cylinder_small`)
- Fixed joint between cutter frames and modelling ground (`FIX_frame_left_ground` and `FIX_frame_right_ground`)
- Cardan joints connecting hydraulic cylinders (`HOO_cylinder_left_frame_left`, `HOO_cylinder_right_frame_right`, `HOO_cylinder_small_frame_right`, `HOO_piston_left_leverl_left`, `HOO_piston_right_lever_right`, and `HOO_piston_small_lever_right`)
- Revolute joints connecting the levers to the frames (`REV_lever_left_frame_left` and `REV_lever_right_frame_right`)

The model had the following nine force components:

- Gravity (gravity)
- Bushings connecting the edge and the levers (`BUS_edge_lever_left` and `BUS_edge_lever_right`)
- Forces representing damping in the hydraulic cylinders (`SFO_cylinder_damping_left` and `SFO_cylinder_damping_right`)

- Forces representing the end stoppers of the hydraulic cylinders (`SFO_cylinder_endstop_left` and `SFO_cylinder_endstop_right`)
- Forces representing the hydraulic force of the hydraulic cylinders (`SFO_cylinder_left` and `SFO_cylinder_right`)

After the joint and force components were defined, the model was ready for simple test simulation, such as computing static equilibrium analysis and simple dynamic simulations. Even though the model does not represent the real system at this phase, it is important to run these test simulations regularly and check that the model behaves reasonably. E.g. this model, if properly modelled, should find successfully static equilibrium so that the hydraulic cylinders are compressed to minimum length and the pistons hit the end stops (the hydraulic pressure forces were not yet modelled at this modelling phase).

The hydraulic system was modelled and simulated in Simulink using Simscape hydraulics library. For the runtime communication of the MSC Adams solver and Simulink, additional model components were created in the mechanical system model to

1) measure hydraulic cylinder lengths and compression speeds, and
2) supply the hydraulic force value to the force components in the mechanical system model.

State variable components were used in MSC Adams for defining the communication interface between the software applications:

- Input signals from Simulink for hydraulic forces (`VAR_cylinder_force_left` and `VAR_cylinder_force_right`)
- Output signals to Simulink as measurements (`VAR_cylinder_length_left`, `VAR_cylinder_length_right`, `VAR_cylinder_velocity_left`, and `VAR_cylinder_velocity_right`)

In Figure 23 is shown a screen image of the definition of the communication interface between MSC Adams and Simulink.

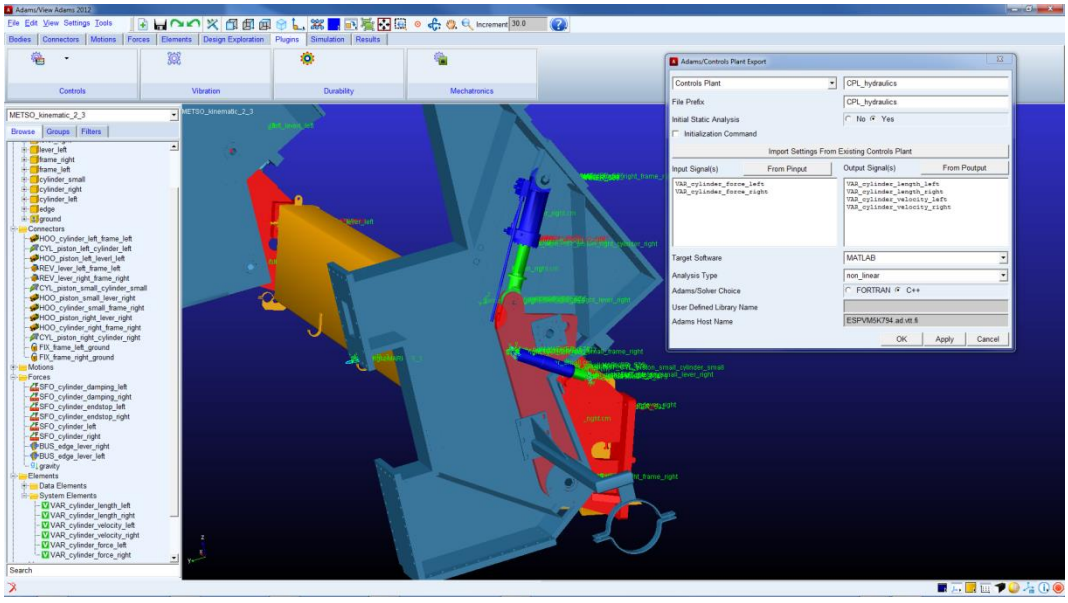


Figure 23: Definitions of the communication interface between MSC Adams solver and Simulink.

It is possible to define the co-simulation mechanism to be FMI instead of the Adams-specific one shown in Figure 23. In that case, the selection for the target software in the Adams/Controls Plant Export dialog would have been FMU. For that selection there would have been an additional option to define the process communication to use TCP/IP instead of PIPE communication. At the time of writing this report, Simulink did not natively support FMI communication mechanism. There are two third party toolboxes available for Matlab/Simulink to add support for FMI:

- FMI Toolbox for Matlab (FMI for model exchange and for co-simulation), by Modelon Ab[19]
- FMI Blockset for Simulink (FMI for co-simulation), by Claytex[20]

The FMI interface was not used in this case, because it was not available in the author's modelling and simulation test environment.

4.2.2 Hydraulic and control system of the test case

The hydraulic, control, and automation systems were modelled and simulated in the Simulink/Simscape environment. The model hierarchy and visual implementation followed the architecture of the systems. In Figure 24 is shown the top level of the overall case system. In this figure, the orange block represents the mechanical subsystem, the magenta block represents the automation and control subsystem, and the green block represents the hydraulic subsystem. The arrows between the blocks represent the output-input signals of the system; most of the signals in this level have physical meaning, such as position, velocity, and force. The control signal between control and hydraulic subsystems is normalised to be between [−1, +1].

The organisation of the subsystems in the Simulink emphasises the modular structure of the virtual prototype of the target system and simplifies the division of the simulation model development for several engineers. In addition, the meaningful interfaces of the subsystems minimise the risk for misunderstandings and errors in the modelling phase and when connecting the sub-models for creating the whole system model.

[19] Modelon FMI Toolbox for Matlab: http://www.modelon.com/products/fmi-toolbox-for-matlab/
[20] Claytex FMI Blockset for Simulink: http://www.claytex.com/products/fmi-blockset-for-simulink/

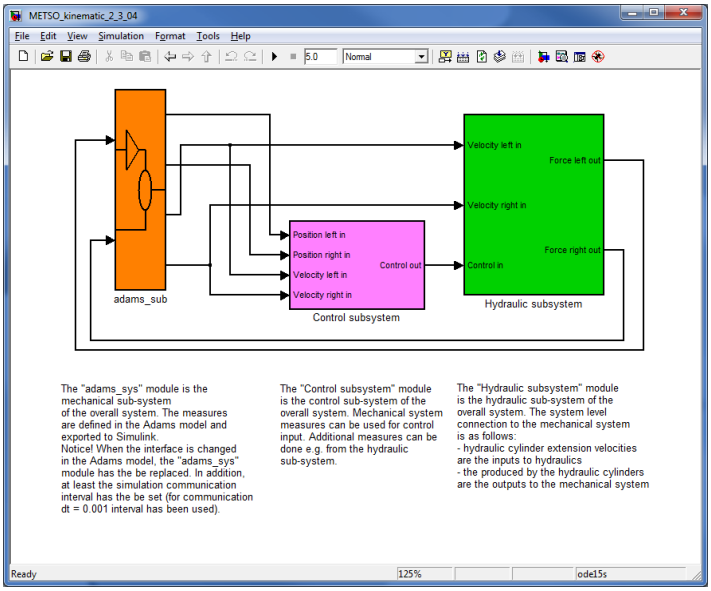


Figure 24: The top level view to the overall test system model. The orange block represents the mechanical subsystem, the pink block represents the automation and control subsystem, and the green block represents the hydraulic subsystem.

The contents of the mechanical subsystem are depicted in Figure 25. The Simulink reference to the mechanical subsystem, i.e. the necessary Simulink model components that connect the external simulation of the mechanical subsystem to the other subsystems modelled in Simulink, is exported from the MSC Adams View pre-processor and does not need to be edited in Simulink. Exporting the Controls Plan model from MSC Adams produces the following files into the modelling directory:

- `<file name>.adm`, a MSC Adams/Solver input file;
- `<file name>.cmd`, a MSC Adams/View command input file (optional);
- `<file name>.m`, a Matlab command input file;
- `<file name>.xmt_txt`, a Parasolid geometry input file (optional);

The optional files are needed if the mechanical model is visualised during the simulation in MSC Adams View pre-processor. The procedure to import the mechanical system model into Simulink is described in detail in the MSC Adams documentation [24]. When the mechanical system model has been imported into Simulink, only the numerical solving and software application communication parameters have to be set. Otherwise, the subsystem model does not need to be changed.

a) b)

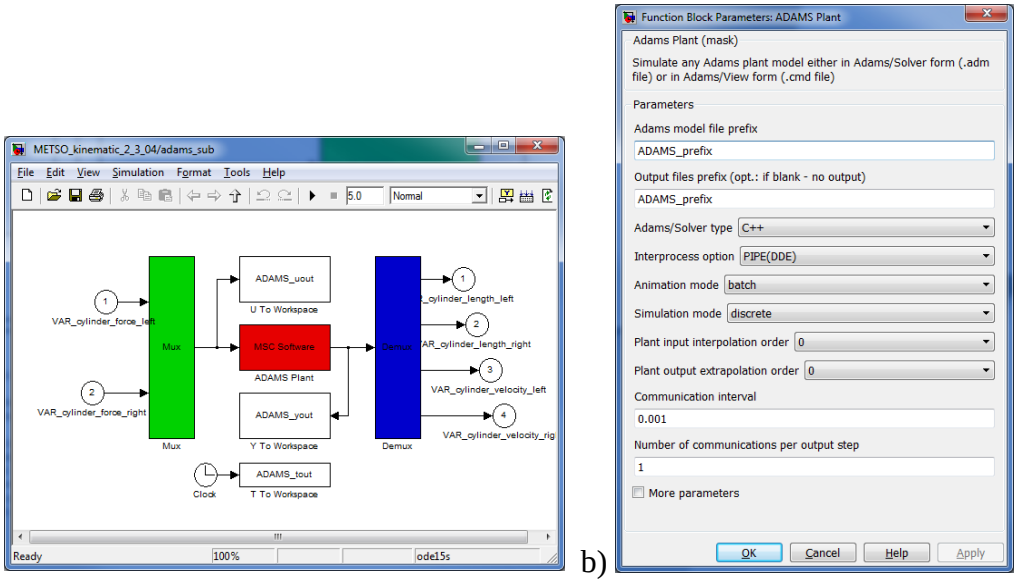


Figure 25: a) The mechanical subsystem model, written from the MSC Adams View pre-processor. b) The numerical solving and software application communication parameters in Simulink.

The automation and control subsystem in the demonstration case is simplified and it is practically a template for a realistic control system model (Figure 26). Despite of the simplicity, the control system model demonstrates the modularity of the virtual prototype architecture and shows how the input and output signals are treated in the model interfaces. In Figure 26, the white icons are the components of the control subsystem, the orange icons represent the input signals to the model of the control subsystem, and the light blue icon represents the control subsystem output signal. The modelled control system does not use the input signals, and only produces time dependent signal for the hydraulic subsystem; the output signal form is presented in Figure 27.

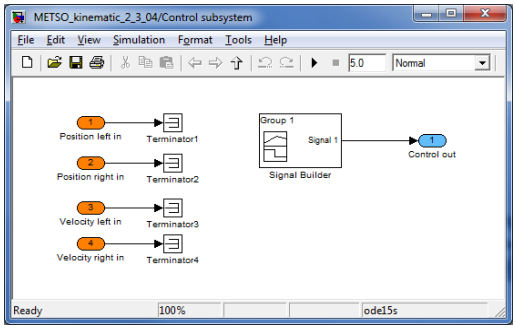


Figure 26: The model of the control subsystem in Simulink. The model is simple but shows the modularity of the overall model and how the signals are treated at the subsystem model interfaces.

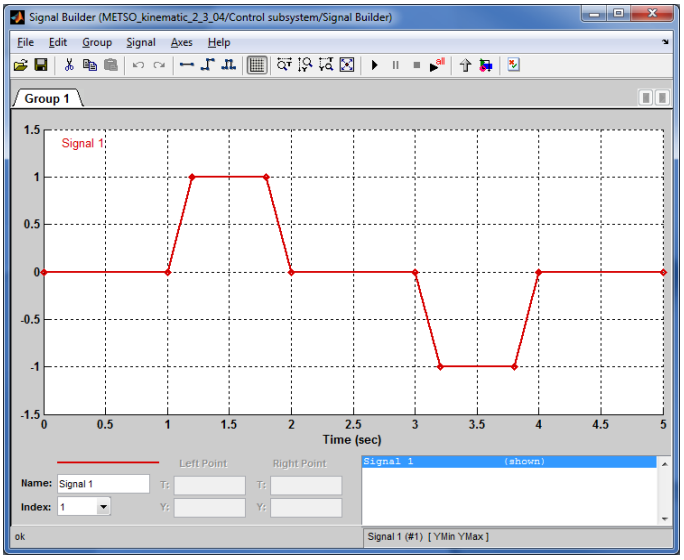


Figure 27: The form of the control subsystem output signal as a function of time.

The hydraulic subsystem that is driving the mechanism in the overall model is modelled and simulated using Matlab/Simscape physical system simulation libraries in Simulink, depicted in Figure 28. In the figure:

- the white icons represent hydraulic components, such as a pump, a valve, cylinders, and pipes;
- the dark yellow icons represent monitor components, used for plotting simulation results during and after a simulation;
- the orange icons represent input signals to the hydraulic subsystem;
- the light blue icons represent the output signals from the hydraulic subsystem; and
- the light grey icons represent lumped model structures that do not have clear physical meaning but are necessary to connect non-physical Simulink signals to physical signals in Simscape in the simulation model (see Figure 29 as an example of lumped model “Cylinder connect left”; see [25] for more information).

The hydraulic subsystem contains the following physical components:

- a hydraulic fluid source (tank);
- an idealised pump;
- a 4/3 directional valve;
- two double-acting hydraulic cylinders;
- a hydraulic fluid tank; and
- hydraulic piping.

The hydraulic pipe walls can be treated either as rigid or flexible.

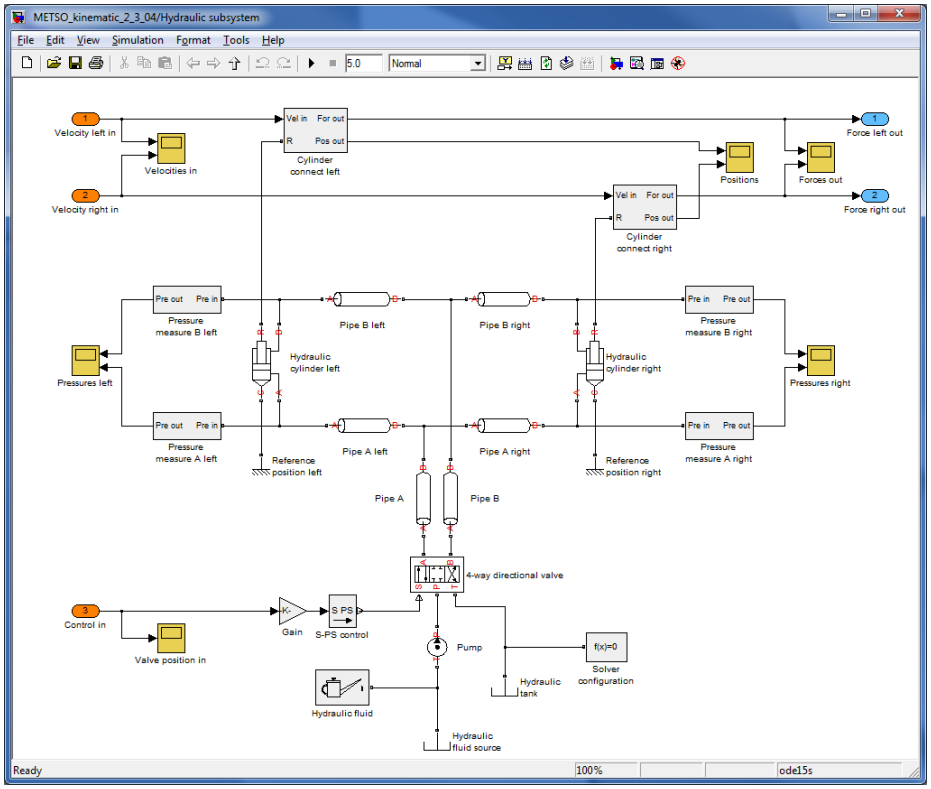


Figure 28: The model of the hydraulic subsystem in Simulink.

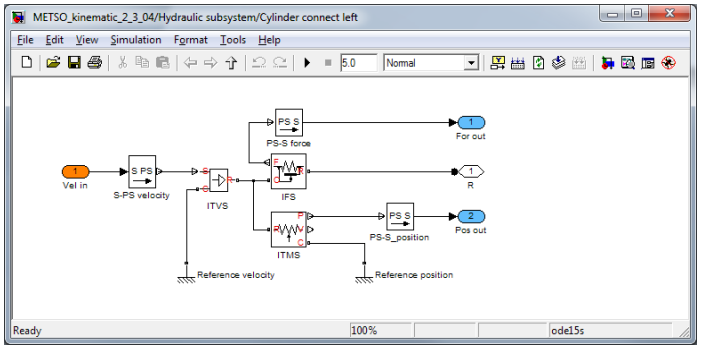


Figure 29: The contents of the "Cylinder connect left" lumped model component. The lumped components in the model of the hydraulic subsystem do not have clear physical counterpart in the real system, but are necessary from the modelling point of view.

4.2.3 Running the overall simulation model

The overall simulation model, including mechanical, automation and control, and hydraulic subsystems, is run using so-called co-simulation approach. In the co-simulation approach, the numerical simulation of the overall system is done using two or more separate numerical solver processes that communicate with each other either after each iteration step of a computational time-step or after a successful computational time-step. If the communication between the solving processes happens after each successful computational time-step, in each solver process, the solutions of the other processes are assumed to be constant during the iterative solving of a time-step. In a case of simulating a continuous coupled system, this approach is an approximation and may lead to qualitatively and quantitatively inaccurate results. The approach is acceptable in many cases due to the advantages

of the approach, such as the simplicity of the modelling process and good enough accuracy for engineering purposes.

The communication between the solving process of the mechanical subsystem, implemented in MSC Adams Solver, and the solving process of the control and hydraulic subsystems, done in Simulink, happens after each successful computational time-step. In other words, during the iterative solving of a time-step of the control and hydraulic subsystems in Simulink, the system states of the mechanical subsystem are assumed to be constant in MSC Adams Solver. To prevent any significant errors in the solution, one millisecond time-stepping was used for the inter-process communication (see Figure 25 b). In the co-simulation with MSC Adams and Simulink, the Simulink process is the master process and the MSC Adams process is the slave process. This means, the Simulink process dictates the time-stepping and calls the MSC Adams process to compute a new one millisecond step and to send the time-step results to Simulink. Both of the numerical solvers are using variable time-steps, which mean that the numerical solver can adjust the size of the time-step according to the transients in the system. The time-stepping is still adjusted to match the communication time-stepping, i.e. the maximum size of a time-step is limited to the size of the communication time-step (i.e. one millisecond in the demonstration case).

4.2.4 Simulation results

Computational simulation of systems, such as the mechanical, hydraulic, and automation and control system presented in the above example, can produce large amount of numerical data. Depending on the user's selections, the data may include the states and their derivatives of the simulated system and any auxiliary data the user has defined. In practice, the data can be e.g. locations, velocities and accelerations of mechanical parts in the system, measured either in the global coordinate system or relative to some other part of the system. In addition, user can define arbitrary functions to be recorded during the simulation, such as damper absorbed power. From the hydraulic and control system models, many measures can be defined, such as pressures or flow rates in different locations of the hydraulic system, or control signal values as a function of time.

Figures 28–31 present output measures plotted from the mechanical system model (simulated with MSC Adams). The measures are from the lifting edge centre point measured relative to the global coordinate system. The location of the measurement point is presented in Figure 30. Figure 31 presents the displacement of the lifting edge relative to the global coordinate system, Figure 32 velocity and Figure 33 acceleration respectively. Figure 34 present a user defined measure, the power absorbed by the motion damper of the mechanical system (the motion damper is shown in Figure 30, the lower cylinder in the left frame). Examples of the simulation results plotted from the hydraulic system (simulated with Simulink) are shown in Figures 32 and 33. In Figure 35 is presented pressures on the cylinder side of the directional valve. Pressures in the right side hydraulic cylinder are presented in Figure 36. The hydraulic system diagram is presented in Figure 28.

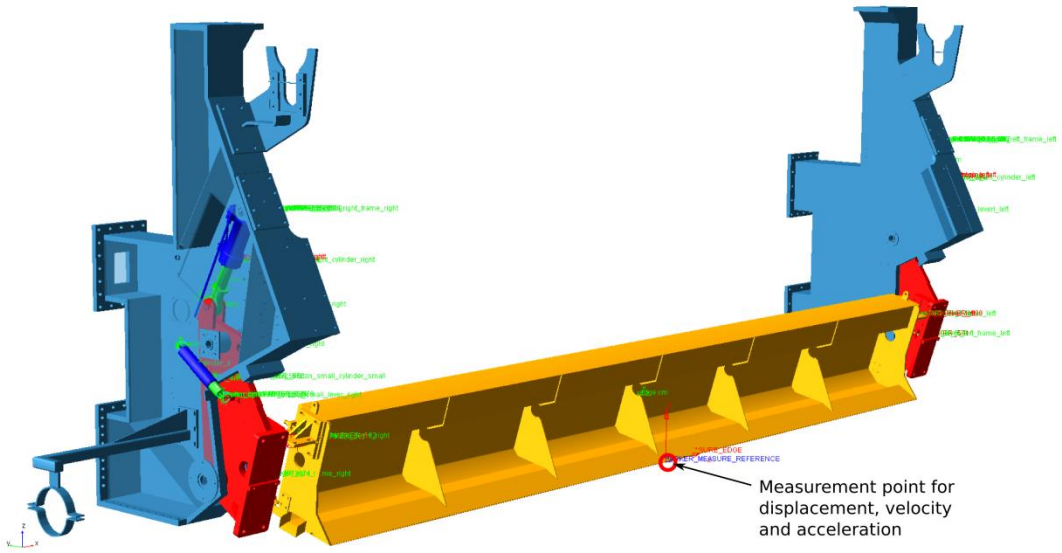


Figure 30: Location of the measurement point for displacement, velocity and acceleration of the lifting edge.

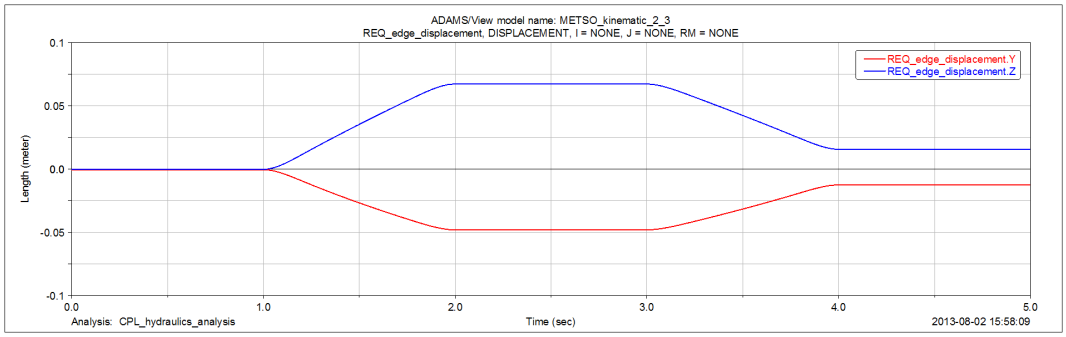


Figure 31: An example of an output measure, the lateral (x coordinate direction in modelling coordinate system) and vertical displacement of the lifting edge.

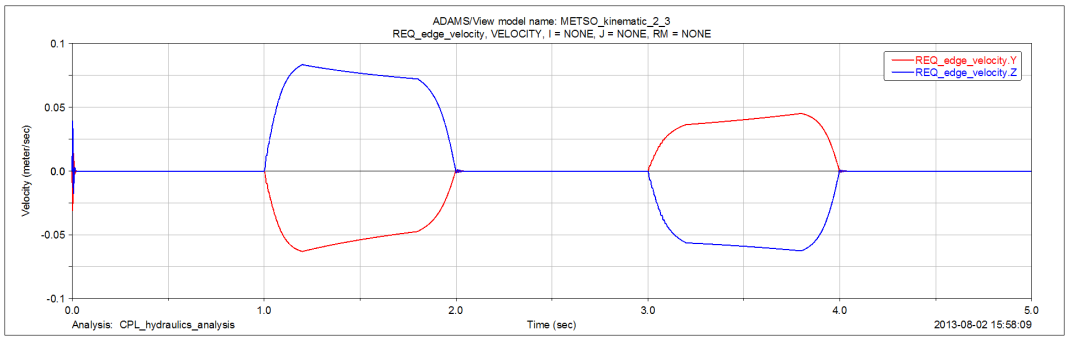


Figure 32: An example of an output measure, the lateral (x coordinate direction in modelling coordinate system) and vertical velocity of the lifting edge.

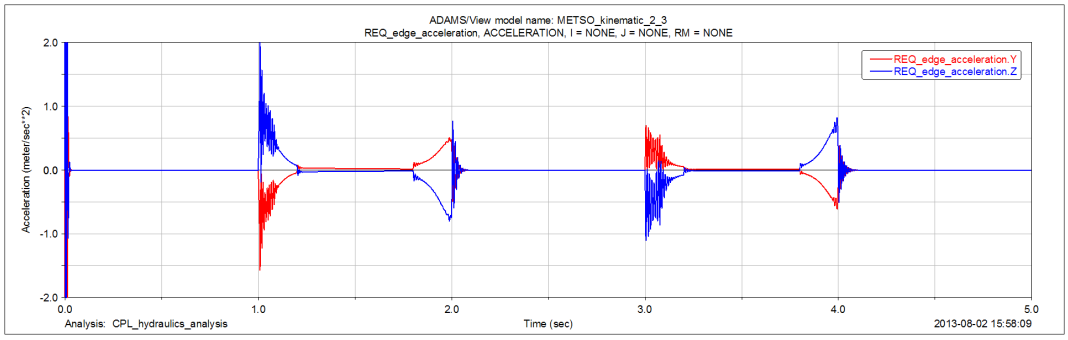


Figure 33: An example of an output measure, the lateral (x coordinate direction in modelling coordinate system) and vertical acceleration of the lifting edge.

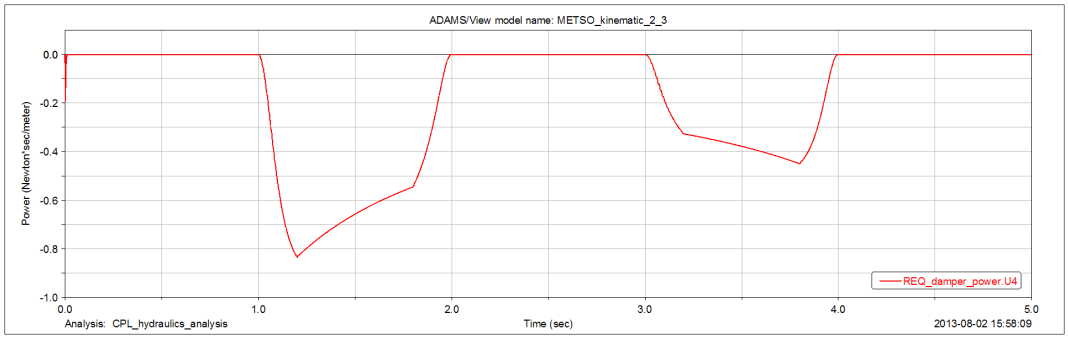


Figure 34: An example of a user defined function expression as an output measure, in this case the power absorbed by the side damper of the system.

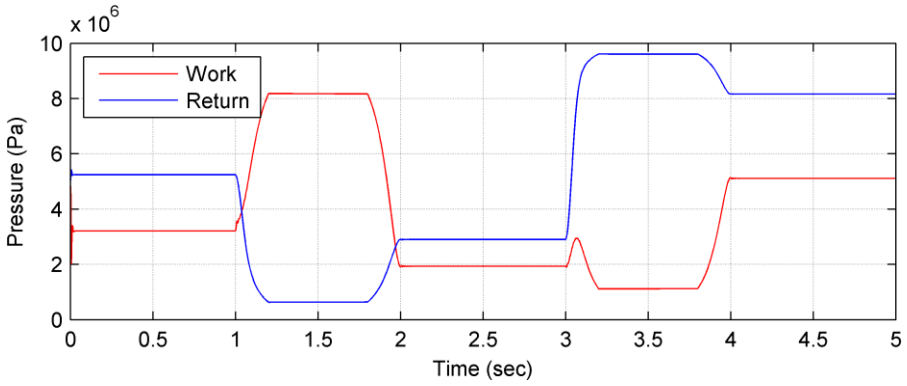


Figure 35: Pressures measured on the hydraulic cylinders' side of the directional valve. "Work" is pressure in the A channel of the valve and "Return" is pressure in the B channel of the valve.

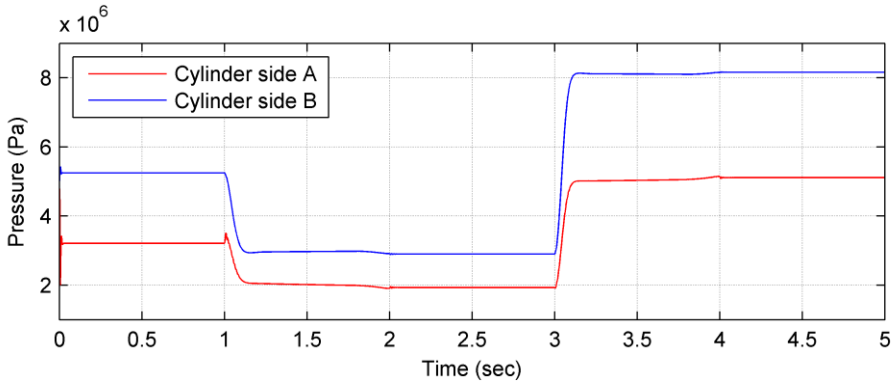


Figure 36: Hydraulic pressures in the right side cylinder. Cylinder side A is the cylinder's extension side and B is the compression side.

5 Conclusions

Simulation of the overall system can be used for machine automation design and research, and the present software tools already support the process, which was demonstrated in the case study described in this report. There are still challenges in data exchange between different software applications, but in most cases these obstacles can be solved. In Section 2.1 of this report, the concept of separating product data from the modelling and simulation tools that are using it was introduced. The vision for the future for this concept is to have an overall product model that combines and links all the relevant product data, including the design data, into one model. With the present software tools this vision is still relatively far, even though there are available integrated design environments that provide tools and data management for design and simulation of many different engineering domains. There is still need for further research and development in this area and the importance of standardisation cannot be overemphasised.

Virtual prototypes that include all the major subsystems of the product or the system can speed up the design process and enable improving the quality of the design. To achieve this, both the software tools and the process of doing the design have to be fitted to operate together. With the present software applications, this requires either designing the process and, based on it, selecting the tools from the offering of many software vendors or selecting one software vendor for providing the overall integration system and then sticking to this choice. The first approach gives more flexibility in selecting the best suited tools for each part of the process but requires understanding of the process and knowledge of the available software tools. The second option is usually more straightforward but it ties the user to one software vendor and may decreasethe room for other options in the selection of software applications in the process. The additional option, i.e. selecting software applications for different parts of the process so that all the software applications integrate fluently together and utilise a common database does not yet exist.

6 Summary

In this report, the use of virtual prototyping and computational product development of multi-technical systems were discussed. The focus was on using multi-technical simulation for automation and control system development and testing.

In the first half of the report, the simulation-based product process and the role of simulation in it was discussed. In addition, the vision of separating the valuable product data (including design and simulation data) was proposed and briefly discussed. The second half of the report focused on the study case of a process to utilise existing CAD model for creating a virtual, simulation-based test environment for automation and control system development and testing. In the beginning of this part, four different approaches for selecting the software applications were discussed. Then, the implementation of the process in the demonstration case, i.e. the modelling of the overall system, running the simulations and using the results, were described and discussed.

The demonstration showed that, at least for the selected case, modelling, simulation and post-processing of a multi-technical simulation system is relatively straightforward and fast with the selected tools. The usefulness and added value of using simulation in product process were discussed already in the introduction of this report. The demonstration gives some understanding of the process for implementing one relatively small multi-technical system but does not give realistic feedback about the challenges in industrial-scale process for large and complex systems' virtual prototyping and related data exchange and data management.

References

[1] Benioff, M. & Lazowska, E. Computational Science: Ensuring America's Competitiveness. President's Information Technology Advisory Committee (PITAC), 2005. 104 p. http://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf (cited on August 1, 2013).

[2] Kortelainen, J. Semantic Data Model for Multibody System Modelling. Doctoral thesis. Espoo: VTT Technical Research Centre of Finland, 2011. 119 p. + 34 p. (VTT Publications 766.) ISBN 978-951-38-7742-2.

[3] ISO 10303-203. Industrial automation systems and integration – Product data representation and exchange – Part 203: Application protocol: Configuration controlled 3D design of mechanical parts and assemblies. Standard, International Organization for Standardization, 2005.

[4] ISO 10303-214. Industrial automation systems and integration – Product data representation and exchange – Part 214: Application protocol: Core data for automotive mechanical design processes. Standard, International Organization for Standardization, 2005.

[5] Manola, F. & Miller, E. RDF Primer. The World Wide Web Consortium, 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ (cited on August 1, 2013).

[6] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F. & Rudolph, S. OWL 2 Web Ontology Language Primer (Second Edition). The World Wide Web Consortium, 2012. http://www.w3.org/TR/2012/REC-owl2-primer-20121211/ (cited on August 1, 2013).

[7] Böhms, M., Leal, D., Graves, H. & Clark, K. Product modelling using Semantic Web technologies. The World Wide Web Consortium, 2009. http://www.w3.org/2005/Incubator/w3pm/XGR-w3pm-20091008/ (cited on August 1, 2013).

[8] Wikipedia, "Windows XP": http://en.wikipedia.org/wiki/Windows_xp (cited on August 1, 2013).

[9] Modelica – A Unified Object-Oriented Language for Physical Systems Modeling. Language Specification, Version 3.2, Revision 1. Modelica Association, 2012. https://www.modelica.org/documents/ModelicaSpec32Revision1.pdf (cited on August 1, 2013).

[10] The website of the Modelica Association: https://www.modelica.org/ (cited on August 1, 2013).

[11] Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Clauß, C., Elmqvist, H., Junghanns, A., Mauss, J., Monteiro, M., Neidhold, T. & others. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In proceedings of 8th International Modelica Conference, Dresden, Germany, 2011. pp. 20–22.

[12] Functional Mock-up Interface for Model Exchange. Technical specification, document version 1.0, MODELISAR Consortium, 2010. https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_ModelExchange_v1.0.pdf (cited on August 1, 2013).

[13] Functional Mock-up Interface for Co-Simulation. Technical specification, document version 1.0, MODELISAR Consortium, 2010.

https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_CoSimulation_v1.0.pdf (cited on August 1, 2013).

[14] FMI PLM Interface – Specification for Product Lifecycle Management (PLM) of modeling, simulation and validation information. Technical specification, document version 1.0, MODELISAR Consortium, 2011. https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_PLM_v1.0.pdf (cited on August 1, 2013).

[15] Functional Mock-up Interface for Model Exchange and Co-Simulation. Technical Specification, document version 2.0 release candidate 1, MODELISAR Consortium, 2013. https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_ModelExchange_and_CoSimulation_v2.0_RC1.pdf (cited on November 18, 2013).

[16] Blochwitz, T. et al. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In proceedings of 9th International Modelica Conference, Munich, Germany, 2012. pp. 173–184.

[17] Paper machine press sections. Technical documentation, Metso Product Vault. Metso Paper, Inc. 2013. http://www.metso.com/MP/Marketing/Vault2MP.nsf/BYWID2/WID-051219-2256E-9A525/$File/Press_sections_PBL.pdf (cited on November 20, 2013).

[18] Bastian, J., Clauß, C., Wolf, S. & Schneider, P. Master for Co-Simulation Using FMI. In proceedings of 8th International Modelica Conference, Dresden, Germany, 2011.

[19] Andersson, C., Åkesson, J., Führer, C. & Gäfvert, M. Import and Export of Functional Mock-up Units in JModelica.org. In proceedings of 8th International Modelica Conference, Dresden, Germany, 2011

[20] Sun, Y., Vogel, S., Steuer, H. & Sector, E. Combining Advantages of Specialized Simulation Tools and Modelica Models using Functional Mock-up Interface (FMI). In proceedings of 8th International Modelica Conference, Dresden, Germany, 2011.

[21] Schubert, C., Neidhold, T. & Kunze, G. Experiences with the new FMI Standard Selected Applications at Dresden University. In proceedings of 8th International Modelica Conference, Dresden, Germany, 2011.

[22] ISO/IEC15288:2008. Systems and software engineering — System life cycle processes. Standard, International Organization for Standardization, 2008.

[23] Haskins, C., Forsberg, K. & Krueger, M. (editors). Systems Engineering Handbook – A Guide for System Life Cycle Processes and Activities. International Council on Systems Engineering (INCOSE), version INCOSE-TP-2003-002-03.1, 2007.

[24] Online documentation for MSC Adams/View, version 2012.2.

[25] Inline documentation for Matlab/Simulink/Simscape, version R2012a (7.14.0.739).

APPENDIX A: List of software applications supporting FMI 1.0

Below is a list of software applications that support FMI version 1.0. The data is copied from the FMI website[21] on July 30, 2013. The meaning of the feature support in the table is as follows:

- Planned: not yet available
- Available: no cross check results submitted
- Verified: passed the cross check

Table 2: FMI support in tools, compatibility table.

Tool supporting FMI	Model exchange		Co-simulation		Notes
	Export	Import	Slave	Master	
Adams		Planned	Available	Available	High end multibody dynamics simulation software from MSC Software
AMESim	Available	Available	Available	Planned	Modelica environment from LMS-Imagine
ANSYS Simplorer		Planned	Planned		ANSYS Simplorer is a multi-domain, multi-technology simulation program from ANSYS.
ASim - AUTOSAR Simulation	Available		Available		AUTOSAR product from Dassault Systèmes
Atego Ace		Available		Available	Co-simulation environment with AUTOSAR and HIL support
@Source	Available				Simulink via @Source
Building Controls Virtual Test Bed				Available	BCVTB is a Software environment, based on Ptolemy II, for co-simulation of, and data exchange with, building energy and control systems.
CATIA	Available	Available	Available	Available	Environment for Product Design and Innovation, including systems engineering tools based on Modelica, by Dassault Systèmes
ControlBuild	Available	Available	Available	Available	Environment for IEC 61131-3 control applications from Dassault Systèmes
CosiMate		Available		Available	Co-simulation Environment from ChiasTek
Cybernetica CENIT		Available		Planned	Industrial product for nonlinear Model Predictive Control (NMPC) from Cybernetica.
Cybernetica ModelFit		Available		Available	Software for model verification, state and parameter estimation, using logged process data. By Cybernetica.
DSHplus	Planned		Planned		Fluid power simulation software from FLUIDON
Dymola	Verified	Available	Verified	Available	Modelica environment from Dassault Systèmes. ModelExchange also available for Simulink using Simulink Coder.
EnergyPlus			Planned	Available	Whole building energy simulation program
FMI Add-in for Excel				Verified	FMI Add-in for Microsoft Excel by Modelon. Offers support for batch simulation of FMUs.
FMI add-on for NI VeriStand		Available			NI VeriStand supports FMI through the use of the FMI add-on for NI VeriStand from Dofware
FMI Blockset for Simulink				Available	Import of FMI Co-Simulation models into Simulink - provided by Claytex.
FMI Library		Verified		Verified	Open source (BSD) C library for integration of FMI technology in custom applications by Modelon.
FMI Target for Simulink Coder			Available		Export of stand-alone FMUs for Co-Simulation from Simulink using Simulink Coder - provided by ITI
FMI Toolbox for CarMaker		Available		Available	For IPG CarMaker via FMI Toolbox for CARMAKER from Modelon.
FMI Toolbox for MATLAB	Verified	Verified	Planned	Verified	FMI Toolbox for MATLAB from Modelon can be used for MATLAB and Simulink.
FMU SDK	Available	Available	Available	Available	FMU Software Development Kit from QTronic.

[21] FMI support in tools, compatibility table: https://www.fmi-standard.org/tools

ICOS "Independent Co-Simulation"		Available	Available	Available	ICOS is a co-simulation tool developed by Virtual Vehicle
JFMI			Available	Available	A Java Wrapper for the Functional Mock-up Interface, based on FMU SDK
JModelica.org	Verified	Verified	Verified	Verified	Open source Modelica environment from Modelon
LMS Virtual.Lab Motion	Planned	Available	Available	Available	Virtual.Lab Motion is a high end multi body software from LMS International
MapleSim	Verified	Planned	Planned	Planned	Modelica-based modeling and simulation tool from Maplesoft
MWorks	Available	Planned	Planned	Planned	Modelica environment from Suzhou Tongyuan
NI LabVIEW		Planned			Graphical programming environment for measurement, test, and control systems from National Instruments
OpenModelica	Available	Available	Planned	Available	Open source Modelica environment from OSMC
OPTIMICA Studio	Verified	Planned	Planned	Planned	Modelica environment from Modelon
Ptolemy II				Planned	Software environment for design and analysis of heterogeneous systems.
PyFMI		Verified		Verified	For Python via the open source package PyFMI from Modelon. Also available as part of the JModelica.org platform.
RecurDyn	Planned	Planned	Planned	Planned	High End Multi Flexible Body Dynamcis Software from FunctionBay
Reference FMUs	Planned		Planned		Reference FMUs supplied by enthusiasts and volunteers to show case specific FMU features
SCADE Display	Planned		Planned		SCADE Display facilitates embedded graphics, display and HMI development and certified code generation for safety-critical displays from ANSYS.
SCADE Suite	Available		Available		SCADE Suite is a model-based development environment with certified code generation for safety critical embedded applications from ANSYS.
Silver	Verified	Verified	Verified	Verified	Virtual integration platform for Software in the Loop from QTronic
SIMPACK	Planned	Available	Planned	Available	High end multi-body simulation software from SIMPACK AG
SimulationX	Verified	Verified	Verified	Verified	Multi-domain simulation tool for design, analysis and virtual prototyping of complex systems by ITI.
SystemModeler	Planned	Planned	Planned	Planned	Modelica environment from Wolfram Research.
TLK FMI Suite		Available		Available	TLK FMI Suite provides LabVIEW and Simulink blocks for FMU simulation
TLK TISC Suite		Available		Available	Co-simulation environment from TLK-Thermo
TWT Co-Simulation Framework			Available	Available	Communication layer tool to flexibly plug together models for performing a co-simulation; front-end for set-up, monitoring and post-processing included
TWT FMU Trust Centre			Available		Cryptographic protection and signature of models including their safe PLM storage; secure authentication and authorization for protected (co-)simulation
xMOD		Available		Available	Heterogeneous model integration environment & virtual instrumentation and experimentation laboratory from IFPEN distributed by D2T.